

POLITECHNIKA GDAŃSKA
INSTYTUT OKRĘTOWY

Włodzimierz J. Martin

PODREČZNIK

PROGRAMOWANIA W JĘZYKU BASIC

I

UŻYTKOWANIA SYSTEMU BASIC/CROOK

do użytku wewnętrznego

Gdańsk 1981

System programowania BASIC-CROOK

Podręcznik programowania
w języku BASIC
i użytkowania systemu

Napisał: W.J. Martin

© W.J. Martin 1981

G d a ń s k 1981

System programowania BASIC-CROOK stanowi znaczne rozszerzenie opracowanych poprzednio translatorów-interpreterów języka BASIC. Przy określaniu specyfikacji i zakresu języka przyjęto jako podstawę specyfikację języka wzorcowego opublikowaną w 1973 roku przez NCC. System BASIC-CROOK daje użytkownikowi co najmniej ten sam zakres możliwości co BASIC wzorcowy wg specyfikacji NCC.

Głównymi autorami systemu programowania BASIC-CROOK są:

Zenon Kapała translator, interpretator, monitor wielo-
dostępny

Włodzimierz J. Martin składnia i specyfikacja języka,
podręcznik, kierowanie całością
prac

Marek Nikodemski obsługa zbiorów, pakiet operacji na
zmiennych indeksowanych

W początkowych pracach brał udział również Sławomir Ilnicki^{z/}.

Wszelkie uwagi na temat systemu oraz wykrytych w nim błędów prosimy kierować wraz z wydrukami itp. do dostawcy systemu lub bezpośrednio do autorów, na adres:

Instytut Okrętowy Politechniki Gdańskiej, Z-d V
ul. Majakowskiego 11/12
80-952 Gdańsk
tel. 47-869

Życzymy przyjemnej pracy pod systemem BASIC.

^{z/} Obecnie w firmie Honeywell - USA

S P I S T R E Ś C I

ZASADY OPISU SKŁADNI	5
ALFABET JĘZYKA BASIC	6
BUDOWA PROGRAMU	7
Komentarze	8
Wiersz końca programu	8
STAŁE, ZMIENNE, WARTOŚCI	9
Wartości arytmetyczne	9
Wartości łańcuchowe	10
Zmienna arytmetyczna prosta	10
Zmienna łańcuchowa prosta	10
Zmienne indeksowane	10
Definicje składniowe zmiennych	11
Stała arytmetyczna	12
Stała łańcuchowa	13
INSTRUKCJA PRZYPISANIA, WYRAŻENIA	14
Wyrażenia arytmetyczne	14
Wyrażenia łańcuchowe	15
Wołanie funkcji standardowej	16
Wyrażenie wskaźnikowe	16
CZYTANIE DANYCH Z PROGRAMU	17
Dane	17
Instrukcja czytania danych	18
Inicjalizacja bloku danych	18
Ustawianie wskaźnika pobierania danych	19
INSTRUKCJE STERUJĄCE	20
Instrukcja zatrzymania	20
Instrukcja skoku	20
Instrukcja skoku ze śladem	20
Instrukcja powrotu według śladu	21
Instrukcja skoku liczonego	23
Instrukcja warunkowa	23

BLOK PĘTLI	26
Instrukcja otwarcia pętli	26
Instrukcja zamknięcia pętli	26
Algorytm wykonywania bloku pętli	27
FUNKCJE DEFINIOWANE	28
Definicja jednowierszowa	29
Definicja wielowierszowa	30
Wiersz końca definicji	31
OPERACJE NA ZMIENNYCH INDEKSOWANYCH	35
Deklaracja wymiarów	36
Wymiary aktualne	37
Instrukcja przypisania zmiennych indeksowanych	33
OPERACJE WEJŚCIA/WYJŚCIA DLA INFORMACJI ZNAKOWEJ	40
Kanał	40
Instrukcja otwarcia kanału	40
Margines	42
Wskaźnik	43
Wskaźnik zbioru	43
Instrukcja wprowadzania	44
Instrukcja wprowadzania wierszami	45
Instrukcja wprowadzania zmiennych indeksowanych	45
Instrukcja wprowadzania wierszami zmiennych indeksowanych	46
Instrukcja drukowania	48
Drukowanie swobodne	48
Drukowanie według wzorca	50
Instrukcja drukowania zmiennych indeksowanych	58
OPERACJE NA ZBIORACH O DOSTĘPIE SWOBODNYM	63
Instrukcja wymazania zawartości zbioru	63
Instrukcja ustawiania wskaźnika zbioru	64
Instrukcja czytania	64
Instrukcja zapisu	65
Instrukcja czytania i zapisu zmiennych indeksowanych	65

SUBPROGRAMY	66
Wiersz końca subprogramu	68
Wiersz nagłówka subprogramu	68
Instrukcja wołania subprogramu	68
INSTRUKCJA RANDOMIZACJI	73
INSTRUKCJA ZMIANY PROGRAMU NAKŁADKOWANIA	75
INSTRUKCJA DOŁĄCZANIA Z BIBLIOTEKI	76
INSTRUKCJA ZANIEDBANIA NADMIARU UJEMNEJ CECHY	83
TABLICA KODÓW ASCII(ISO-7)	84
FUNKCJE STANDARDOWE JĘZYKA BASIC	85
Podstawowe funkcje standardowe arytmetyczne	85
Inne funkcje standardowe arytmetyczne	86
Funkcje standardowe arytmetyczne dotyczące wartości łańcuchowych	88
Funkcje standardowe arytmetyczne dotyczące zmiennych indeksowanych	89
Funkcje standardowe arytmetyczne systemowe	90
Funkcje standardowe łańcuchowe	92
Funkcje standardowe łańcuchowe systemowe	95
KONWERSACJA UŻYTKOWNIKA Z SYSTEMEM BASIC-EXT	96
URUCHOMIENIE SYSTEMU BASIC	97
ZLECENIA SYSTEMU BASIC	98
WSPÓŁPRACA Z SYSTEMEM OPERACYJNYM CROOK-3	109
Uruchomienie maszyny i systemu ,	109
Wpisywanie się do systemu	111
Niektóre zlecenia systemu operacyjnego	112
Zatrzymywanie i wznowianie pracy systemu BASIC	119
Niektóre komunikaty systemu operacyjnego	119
Niektóre programy codziennego użytku	123
WERSJA BEZDYSKOWA SYSTEMU BASIC	126

ZASADY OPISU SKŁADNI

W niniejszym podręczniku zastosowano uproszczony sposób opisu składni instrukcji i zleceń, którego podstawowe reguły są opisane poniżej:

1. W nawiasy kątowe $\langle \rangle$ ujęte są obiekty definiowane - bądź oddzielną definicją bądź opisowo.
2. Napis $::=$ oznacza "równoważne z definicji".
3. Napisy zamknięte w nawiasach kwadratowych $[]$ są opcjonalne, tj. można je pominąć.
4. Napisy zawarte w klamrach $\{ \}$ są obowiązkowe, tj. muszą wystąpić. Jeżeli w klamrach lub w nawiasach kwadratowych występuje kilka napisów pod sobą - należy za każdym razem wybrać tylko jeden z nich.
5. Trzy kropki za nawiasem kwadratowym lub klamrą oznaczają możliwość powtarzania zawartości klamr lub nawiasów kwadratowych (bądź jednego, za każdym razem dowolnego, z napisów umieszczonych jeden pod drugim).
6. Napisy nie ujęte w nawiasy kątowe oraz nie będące znakami opisanymi wyżej reprezentują same siebie.

Ze względów technicznych napisy definiujące składnię są niekiedy umieszczone w dwóch wierszach - należy je wtedy traktować tak, jakby znajdowały się w jednym wierszu. Terminy wewnątrz nawiasów zapisywane są często skrótami, na ogół łatwymi do odcyfrowania. W szczególności

$\langle w.a. \rangle$ należy czytać jak $\langle \text{wyrażenie arytmetyczne} \rangle$
 $\langle w.w. \rangle$ należy czytać jak $\langle \text{wyrażenie wskaźnikowe} \rangle$

ALFABET JĘZYKA BASIC

W instrukcjach języka BASIC używane są następujące znaki:

Litery	@ABCDEFGHIJKLMNOPQRSTUVWXYZ
cyfry	0 1 2 3 4 5 6 7 8 9
kropka	.
separatory	; , :
znaki relacji	> < =
znaki operacji arytmetycznych	+ - * / ↑
znak waluty	\$ lub £ lub ¤
znak numeru	#
nawiasy	()
cudzysłów	"
apostrof	'
znak operacji złożenia	&

W stałych łańcuchowych i komentarzach mogą występować również dowolne inne znaki.

W wartościach łańcuchowych przetwarzanych przez programy BASIC mogą występować wszystkie znaki ISO-7 ASCII o kodach 1 do 127, tzn. również znaki nie mające reprezentacji graficznej. Wyjątek stanowią znaki NUL (kod 0) i DC4 (kod 20). Znaki NUL są przez system BASIC usuwane. Znak DC4 jest natomiast używany jako znacznik końca zbioru. Jego wyprowadzenie do zbioru instrukcjami wyjścia może zatem spowodować niezamierzone przez programistę efekty.

BUDOWA PROGRAMU

Program w języku BASIC składa się z programu głównego i subprogramów (procedur), przy czym te ostatnie mogą nie występować. Subprogramy stanowią oddzielne jednostki programowe, w których wszystkie obiekty są lokalne, a komunikacja z programem głównym (lub programem wołającym) odbywa się wyłącznie za pośrednictwem parametrów.

Program (lub subprogram) składa się z wierszy, rozpoczynających się numerami. Numer wiersza jest liczbą naturalną mniejszą od 32768. Program wykonuje się w kolejności wzrastających numerów wiersza, począwszy od wiersza o najmniejszym numerze, przy czym wiersze niewykonywane są omijane. Kolejność natomiast wprowadzania wierszy do pamięci jest dowolna.

Tworzenie programu (wprowadzanie wierszy) odbywa się w trybie programowym, w którym system zgłasza nasłuch znakiem gwiazdki * dla zbioru roboczego - programu głównego, zaś znakiem numeru # dla podzbioru roboczego - subprogramu. Przejście z kontekstu jednej jednostki programowej do innej odbywa się za pomocą zlecenia NAME - patrz opis zleceń. Przy pracy w trybie programowym spacje są przez system ignorowane, za wyjątkiem komentarzy i wartości łańcuchowych.

Każdy wprowadzony wiersz programu jest przez system sprawdzany na poprawność składni. Po stwierdzeniu poprawności jest umieszczany w pamięci (w zbiorze lub podzbiorze roboczym), stając się częścią tworzonego programu lub subprogramu. Stwierdzenie błędu składni powoduje wydrukowanie komunikatu o błędzie i odrzucenie wiersza, bez żadnego wpływu na tworzony program lub subprogram. Wprowadzenie poprawnego wiersza programu o takim samym numerze jak już istniejący w zbiorze roboczym (programie lub subprogramie) powoduje usunięcie z programu wiersza istniejącego i umieszczenie w jego miejsce wiersza właśnie wprowadzonego. Wprowadzenie samego numeru wiersza usuwa znajdujący się w programie wiersz o tym numerze.

Znakiem kończącym wiersz jest znak zmiany wiersza LF, o kodzie 10.

Wiersze zawierające instrukcje programu mają następującą budowę:

<nr wiersza> <instrukcja> [<komentarz>]

gdzie <komentarz> jest dowolnym ciągiem znaków.

Znaki komentarza, mimo że są umieszczane w pamięci, nie mają zupełnie znaczenia przy wykonywaniu programu.

Komentarze można również umieścić w programie w specjalnie do tego celu przeznaczonych wierszach o następującej budowie:

<nr wiersza> REM [<komentarz>]

Wiersze takie są wierszami niewykonywanymi, tzn. są przy wykonywaniu programu ignorowane.

Wiersz końca programu

Każdy program główny musi być zakończony wierszem końca programu o następującej budowie:

<nr wiersza> END [<komentarz>]

Wiersz końca programu wypełnia dwie funkcje:

- a. stanowi dla translatora informację o kontekstowym końcu programu. W momencie uruchamiania programu wiersze o numerach wyższych od numeru wiersza końca programu nie są uważane przez system BASIC za należące do programu;
- b. napotkanie wiersza końca programu w czasie wykonywania programu powoduje zakończenie wykonywania, z odpowiednim komunikatem i przejściem na nasłuch w trybie programowym.

STAŁE, ZMIENNE, WARTOŚCI

Programy w języku BASIC wykonują operacje na wartościach dwóch typów: wartościach arytmetycznych i wartościach łańcuchowych.

Wartości arytmetyczne są liczbami rzeczywistymi odpowiadającymi np. typowi real w Algolu, liczby całkowite są przez system traktowane również tak samo jak liczby rzeczywiste. Zakres i dokładność przedstawiania liczb związane są ściśle z ich reprezentacją w maszynie. Odpowiednich wartości granicznych można używać w programach, korzystając z funkcji standardowych INF, EPS, NIL i INX.

W maszynie MERA-400 wartości liczb muszą spełniać zawsze jedną z poniższych zależności:

$$\begin{aligned} 2^{-129} &\leq \text{liczba} \leq (1 - 2^{-39}) \cdot 2^{127} \\ \text{liczba} &= 0 \\ -1 \cdot 2^{127} &\leq \text{liczba} \leq \left(-\frac{1}{2} - 2^{-39}\right) \cdot 2^{128} \end{aligned}$$

czyli w przybliżeniu wartość bezwzględna liczb musi zawierać się w przedziale

$$2.94 \cdot 10^{-39} \dots 1.7 \cdot 10^{38}$$

lub musi być równa zero.

Jeżeli w wyniku wykonywania działań arytmetycznych nastąpi wyjście poza podane wyżej zakresy wartości - system zasygnalizuje przekroczenie zakresu liczb, tj. nadmiar cechy i wykonywanie programu zostanie przerwane.

Największą liczbą całkowitą przedstawioną dokładnie w maszynie MERA-400 przy pracy w systemie BASIC jest liczba $2^{39} - 1 = 549,755,813,887$. Większe liczby całkowite nie mają dokładnej reprezentacji maszynowej, a jedynie przybliżoną.

Wartości łańcuchowe są ciągami znaków ASCII (ISO-7) o dowolnej długości. Długości (liczby znaków) wartości łańcuchowych ograniczone są jedynie wielkością dostępnej programowi w danej chwili pamięci.

Do umożliwienia manipulowania wartościami w programach służą zmienne. Zmienną można wyobrazić sobie jako komórkę pamięci, zdolną do przechowania jednej wartości. Ponieważ w języku BASIC można wykonywać operacje na wartościach dwóch typów - arytmetycznych i łańcuchowych - również i zmienne są takich samych dwóch typów.

Zmienna arytmetyczna prosta jest oznaczana w programach pojedynczą literą albo literą i cyfrą:

$\langle \text{zmienna arytmetyczna prosta} \rangle ::= \langle \text{litera} \rangle [\langle \text{cyfra} \rangle]$

Można zatem w programie użyć $27 \times 11 = 297$ różnych zmiennych arytmetycznych prostych (znak @ jest również traktowany jak litera).

Zmienna łańcuchowa prosta oznaczona jest w programach literą i znakiem waluty, albo literą, cyfrą i znakiem waluty:

$\langle \text{zmienna łańcuchowa prosta} \rangle ::= \langle \text{litera} \rangle [\langle \text{cyfra} \rangle] \langle \text{znak waluty} \rangle$

przykładami zmiennych arytmetycznych prostych będą:

A X Z3 B5 C0 @3

zaś zmiennych łańcuchowych prostych:

C\$ A\$ F3\$ Y0\$ @7\$ Q1\$

Oprócz zmiennych prostych mogą w programach BASIC występować zmienne indeksowane o jednym lub dwóch wskaźnikach, tj. odpowiednio wektory i macierze. Element zmiennej indeksowanej arytmetycznej jest zmienną arytmetyczną, tj. ma wartość arytmetyczną, podobnie jak zmienna arytmetyczna prosta.

Tak samo każdy element zmiennej indeksowanej łańcuchowej przechowuje wartość łańcuchową czyli ciąg znaków ASCII o dowolnej długości.

Zmienne indeksowane, podobnie jak zmienne proste, nie wymagają w języku BASIC oddzielnej deklaracji typu. Wystąpienie w programie elementu zmiennej indeksowanej spowoduje zarezerwowanie przez system obszaru pamięci, pozwalającego na przechowywanie takiej liczby elementów, by wskaźnik (wskaźniki) zmiennej indeksowanej mogły przyjmować wartości od zera do dziesięciu. Zatem wektor będzie miał 11 elementów, zaś macierz 121 elementów.

Nazwą zmiennej indeksowanej arytmetycznej jest pojedyncza litera, nazwą zmiennej indeksowanej łańcuchowej - litera i znak waluty. Taka sama nazwa (litera lub litera i znak waluty) może oznaczać zmienną prostą i zmienną indeksowaną, nie może natomiast oznaczać jednocześnie wektora i macierzy. Elementy zmiennych indeksowanych oznaczają się w programie przez podanie nazwy, za którą w nawiasach podaje się wskaźnik lub w przypadku macierzy - dwa wskaźniki rozdzielone przecinkiem.

Poniżej podane są definicje składniowe zmiennych.

$$\langle \text{wektor} \rangle ::= \left\{ \begin{array}{l} \langle \text{wektor arytmetyczny} \rangle \\ \langle \text{wektor łańcuchowy} \rangle \end{array} \right\}$$
$$\langle \text{macierz} \rangle ::= \left\{ \begin{array}{l} \langle \text{macierz arytmetyczna} \rangle \\ \langle \text{macierz łańcuchowa} \rangle \end{array} \right\}$$
$$\left\langle \begin{array}{l} \text{zmienna indeksowa} \\ \text{arytmetyczna} \end{array} \right\rangle ::= \left\{ \begin{array}{l} \langle \text{wektor arytmetyczny} \rangle \\ \langle \text{macierz arytmetyczna} \rangle \end{array} \right\}$$
$$\left\langle \begin{array}{l} \text{zmienna indeksowa} \\ \text{łańcuchowa} \end{array} \right\rangle ::= \left\{ \begin{array}{l} \langle \text{wektor łańcuchowy} \rangle \\ \langle \text{macierz łańcuchowa} \rangle \end{array} \right\}$$
$$\left\langle \begin{array}{l} \text{wektor} \\ \text{arytmetyczny} \end{array} \right\rangle ::= \langle \text{litera} \rangle$$

$\langle \text{macierz arytmetyczna} \rangle ::= \langle \text{litera} \rangle$

$\langle \text{wektor łańcuchowy} \rangle ::= \langle \text{litera} \rangle \langle \text{znak waluty} \rangle$

$\langle \text{macierz łańcuchowa} \rangle ::= \langle \text{litera} \rangle \langle \text{znak waluty} \rangle$

$\langle \text{element wektora} \rangle ::= \langle \text{wektor} \rangle (\langle \text{wyrażenie wskaźnikowe} \rangle)$

$\langle \text{element macierzy} \rangle ::= \langle \text{macierz} \rangle (\langle \text{wyr. wskaźn.} \rangle, \langle \text{wyr. wskaźn.} \rangle)$

$\langle \text{zmienna arytmetyczna} \rangle ::= \left\{ \begin{array}{l} \langle \text{zmienna arytmetyczna prosta} \rangle \\ \langle \text{element wektora arytmetycznego} \rangle \\ \langle \text{element macierzy arytmetycznej} \rangle \end{array} \right\}$

$\langle \text{zmienna łańcuchowa} \rangle ::= \left\{ \begin{array}{l} \langle \text{zmienna łańcuchowa prosta} \rangle \\ \langle \text{element wektora łańcuchowego} \rangle \\ \langle \text{element macierzy łańcuchowej} \rangle \end{array} \right\}$

$\langle \text{zmienna} \rangle ::= \left\{ \begin{array}{l} \langle \text{zmienna arytmetyczna} \rangle \\ \langle \text{zmienna łańcuchowa} \rangle \end{array} \right\}$

Przykładami elementów zmiennych indeksowanych mogą być np.

A(8) C(2*X,0) Z\$(1,5) X\$(N+3)

W momencie uruchamiania programu wszystkie zmienne są inicjalizowane: zmiennym arytmetycznym nadawana jest wartość zero, a zmiennym łańcuchowym wartość łańcucha pustego.

Stałe w języku BASIC-EXT są, podobnie jak zmienne, dwóch typów - arytmetyczne i łańcuchowe.

Stałe arytmetyczne są liczbami, zapisanymi za pomocą znaków ASCII w systemie dziesiętnym. Można je zapisywać jako liczby całkowite, liczby ułamkowe z kropką dziesiętną oraz w postaci półlogarytmicznej z zastosowaniem litery E, mającej znaczenie "razy dziesięć do potęgi".

$\langle \text{stała arytmetyczna} \rangle ::= \langle \text{liczba} \rangle$

$$\langle \text{liczba} \rangle ::= \left[\begin{array}{l} + \\ - \end{array} \right] \left\{ \begin{array}{l} \langle \text{cyfra} \rangle [\langle \text{cyfra} \rangle] \dots [.\langle \text{cyfra} \rangle] \dots \\ [\langle \text{cyfra} \rangle] . \langle \text{cyfra} \rangle [\langle \text{cyfra} \rangle] \dots \end{array} \right\} \left[\begin{array}{l} E \\ \end{array} \right] \langle \text{liczba całkowita} \rangle$$

$$\langle \text{liczba całkowita} \rangle ::= \left[\begin{array}{l} + \\ - \end{array} \right] \langle \text{cyfra} \rangle [\langle \text{cyfra} \rangle] \dots$$

Liczbę 1000 można zapisać w języku BASIC-EXT np. następująco:

1000 +1000 1E3 +.1E+4 10E2

Wszystkie powyższe zapisy są poprawne i równoważne.

Znaki tworzące liczbę można rozdzielać pojedynczymi spacjami.

Stałe łańcuchowe zapisywane są w programach BASIC w postaci łańcuchów zamkniętych, tj. ciągów znaków ASCII zamkniętych w znaki cudzysłowu.

$$\langle \text{stała łańcuchowa} \rangle ::= "[\langle \text{znak ASCII} \rangle] \dots"$$

Napis "" jest poprawną stałą i oznacza łańcuch pusty, o długości zero.

Wszystkie spacje znajdujące się wewnątrz łańcucha zostają zachowane w taki sposób, w jaki zostały zapisane.

INSTRUKCJA PRZYPISANIA. WYRAŻENIA

Przypisanie to nadanie zmiennej wartości, ze zniszczeniem wartości poprzedniej. Przypisania mogą być dwóch typów, odpowiednio do dwóch typów wartości w języku BASIC.

$$\langle \text{przypisanie} \rangle ::= \left\{ \begin{array}{l} \langle \text{przypisanie arytmetyczne} \rangle \\ \langle \text{przypisanie łańcuchowe} \rangle \end{array} \right\}$$
$$\langle \text{przypisanie łańcuchowe} \rangle ::= \langle \text{zmienna łańcuch.} \rangle [= \langle \text{zmienna łańcuch.} \rangle] \dots \\ = \langle \text{wyrażenie łańcuch.} \rangle$$
$$\langle \text{przypisanie arytmetyczne} \rangle ::= \langle \text{zmienna arytmetyczna} \rangle [= \langle \text{zmienna arytmetyczna} \rangle] \dots \\ = \langle \text{wyrażenie arytmetyczne} \rangle$$

W wyrażeniach arytmetycznych mogą występować zmienne arytmetyczne, stałe arytmetyczne orazwołania funkcji arytmetycznych standardowych albo definiowanych; operatory działań arytmetycznych oraz nawiasy, zmieniające porządek wykonywania działań.

Znaki operatorów arytmetycznych są następujące od najwyższego do najniższego priorytetu :

zmiana znaku	
↑ potęgowanie	
* mnożenie	/ dzielenie
+ dodawanie	- odejmowanie

W czasie obliczania wartości wyrażenia działania wykonywane są według priorytetu, od lewej do prawej, z uwzględnieniem nawiasów zmieniających kolejność wykonywania działań. Dwa operatory nie mogą wystąpić obok siebie.

Oto przykłady wyrażen arytmetycznych BASIC-u i ich odpowiedników w zwykłej notacji matematycznej:

1/2/A

$$\frac{1}{2a}$$

SQR(1-SIN(X)^2)

$$1 - \sin^2 x$$

A*A*(B-1)/C(I)/LOG(1-EXP(X))

$$\frac{a^2(b-1)}{C_i \ln(1 - e^x)}$$

Należy pamiętać o tym, że maszyna liczy ze skończoną dokładnością, tak że np. X^2 da na ogół inny wynik niż $X \times X$.

W wyrażeniach łańcuchowych mamy do czynienia z jednym tylko operatorem - operatorem złożenia (konkatenacji). Operandami mogą być stałe łańcuchowe, zmienne łańcuchowe oraz wołania funkcji łańcuchowych standardowych i definiowanych.

Działanie operatora złożenia jest takie, że z dwóch wartości łańcuchowych stojących po jego lewej i prawej stronie tworzy jedną wartość łańcuchową.

Na przykład wartość wyrażenia

"ABC"&"DEFG"

jest taka sama jak stałej "ABCDEFGG".

Ponizej podane są przykłady wyrażeń łańcuchowych; wyrażenia w tym samym wierszu mają taką samą wartość.

```

A&C$(Z+3)&"***"
"--"&GAP$(4)&"--"      "--"&MUL$(2,"")&"--"
MUL$(5,CHR$(65))        "AAAAA"
"LAB"&CHR$(48+J)
"VAR"&SDL$(STR$(K),CHR$(32))

```

Główne możliwości przetwarzania wartości łańcuchowych daje bogaty zestaw funkcji standardowych.

Wołanie funkcji standardowej składa się z nazwy funkcji, za którą w nawiasach znajdują się parametry aktualne, rozdzielone przecinkami. Parametry aktualne są wyrażeniami, ich liczba, kolejność i typ muszą być odpowiednie dla danej funkcji. Nazwy funkcji standardowych arytmetycznych składają się zawsze z trzech liter, nazwy funkcji standardowych łańcuchowych - z trzech liter i znaku waluty. Przykłady wołań występują w podanych poprzednio przykładach wyrażeń arytmetycznych i łańcuchowych.

Nazwy, parametry i znaczenie wszystkich 67 funkcji standardowych języka BASIC-EXT podane są przy końcu podręcznika.

Wyrażenie wskaźnikowe (indeks) ma taką samą składnię jak wyrażenie arytmetyczne. Różnica jest taka, że w czasie wykonywania programu system po obliczeniu wartości wyrażenia wydziela z niego część całkowitą tj. bierze największą liczbę całkowitą nie większą od obliczonej wartości i tę traktuje jako wartość wyrażenia wskaźnikowego. Działanie tego mechanizmu jest takie samo, jak działanie funkcji standardowej INT(X).

Wiersz programu, zawierający instrukcję przypisania ma, podobnie jak i inne wiersze zawierające instrukcje, następującą budowę:

```
<nr wiersza> <instrukcja przypisania> [ <komentarz> ]  
<instrukcja przypisania> ::= LET <przypisanie> [ ; <przypisanie> ] ...
```

Jest to jedyny w języku BASIC przypadek, gdzie można w instrukcji pominąć jej słowo kluczowe - w tym przypadku LET. Jest ono jednak przez system zawsze wstawiane przy drukowaniu programu, umieszczaniu programu w zbiorze itp.

Poniżej - przykłady wierszy programu zawierających instrukcję przypisania:

Wykonywanie instrukcji przypisania jest następujące:

- poszczególne przypisania, rozdzielone średnikami, wykonywane są kolejno, od lewej do prawej strony;
- w każdym przypisaniu obliczana jest wartość wyrażenia, znajdującego się po prawej stronie ostatniego znaku równości. Następnie wartość tego wyrażenia nadawana jest jednocześnie wszystkim zmiennym.

Grupowanie kilku przypisań w jednej instrukcji przypisania pozwala zaoszczędzić zarówno miejsca w pamięci, jak i czasu w czasie wykonywania programu. Należy jednak używać go rozważnie, gdyż chcąc np. zmienić jedno tylko przypisanie w takim wierszu, będziemy musieli wprowadzić od nowa cały wiersz, tracąc w ten sposób swój czas.

Grupowanie powinno dotyczyć przypisań logicznie związanych i nie pogarszać czytelności programu, której znaczenia nigdy dość podkreślać.

CZYTANIE DANYCH Z PROGRAMU

Język BASIC pozwala umieścić w programie dane, pogrupowane w odpowiedni sposób. W czasie wykonywania programu dane te są pobierane odpowiednią instrukcją, z nadawaniem ich wartości zmiennym.

Dane umieszcza się w wierszach o następującej budowie:

$\langle \text{nr wiersza} \rangle \text{ DATA } \langle \text{dana} \rangle [, \langle \text{dana} \rangle] \dots [\langle \text{komentarz} \rangle]$

$$\langle \text{dana} \rangle ::= \begin{cases} \langle \text{stała arytmetyczna} \rangle \\ \langle \text{stała łańcuchowa} \rangle \\ \langle \text{łańcuch otwarty} \rangle \end{cases}$$

Jak widać z powyższego, dane arytmetyczne zapisywane są w postaci liczb, zaś dane łańcuchowe w postaci łańcucha, zamkniętego w znaki cudzysłowu, bądź otwartego tzn. ciągu

znaków ASCII. Stosowanie zapisu w postaci łańcucha otwartego pozwala uniknąć pisania cudzysłowu, ale nie pozwala umieścić wewnątrz łańcucha przecinka, spacji itp. Dozwolony jest również łańcuch otwarty pusty.

W przypadkach wątpliwych należy zawsze używać łańcuchów zamkniętych (stałych łańcuchowych). Wiersz z danymi jest wierszem niewykonywanym i może być umieszczony w dowolnym miejscu programu. Wszystkie wiersze danych danej jednostki programowej, system porządkuje według wzrastających numerów wierszy w jeden zestaw danych.

Czytanie danych w czasie wykonywania programu odbywa się za pomocą jednej z form instrukcji czytania, o następującej postaci:

$$\text{READ} \left\{ \begin{array}{l} \langle \text{zmienna} \rangle \\ \langle \text{funkcja definiowana} \rangle \end{array} \right\} \left[\begin{array}{l} \langle \text{zmienna} \rangle \\ \langle \text{funkcja definiowana} \rangle \end{array} \right] \dots$$

Pierwsza wykonana instrukcja czytania pobiera pierwszą daną z całego bloku i jej wartość nadaje pierwszej zmiennej lub funkcji z listy, następnie drugą daną, której wartość nadaje drugiej zmiennej z listy instrukcji czytania itd. aż do wyczerpania listy zmiennych.

Następna instrukcja czytania pobierze następną kolejną, jeszcze nie użytą daną z bloku danych. Jeżeli blok danych wyczerpie się przed wyczerpaniem listy, system zasygnalizuje błąd. Funkcje definiowane mogą wystąpić na liście tylko wewnątrz bloku swojej definicji.

Typy danych i zmiennych muszą być odpowiednio zgodne.

Instrukcja o postaci:

RESET

powoduje ustawienie wskaźnika pobierania danych na pierwszą daną bloku, tj. inicjalizację bloku danych.

Instrukcja o postaci:

RESET # $\emptyset$: $\langle \text{wyrażenie wskaźnikowe} \rangle$

spowoduje w czasie wykonywania programu ustawienie wskaźnika pobierania danych na n-tą daną w bloku, gdzie n jest wartością wyrażenia wskaźnikowego. Pierwsza dana bloku danych ma numer 0.

Przykładowo, jeżeli blok danych jednostki programowej składa się z następujących wierszy

```
100 DATA 1,2,3,4,5
110 DATA 6,7,8 ' TO SA 3 DANE ARYTM.
500 DATA A,B,C,"12",3 ' "12" TO DANA LACUCH.
600 DATA "X Y Z"
```

to wykonanie poniższego fragmentu programu nada poszczególnym zmiennym następujące wartości:

```
200 READ X1,X2,X3,X4,X5,X6
210 READ X7,X8
220 RESET #0: 4 ' USTAWIENIE NA 5. DANA
230 READ Y5,Y6,Y7,Y8,A$,B$
240 READ C$,L6$,N1,K5$
```

X1 = 1	Y6 = 6
X2 = 2	Y7 = 7
X3 = 3	Y8 = 8
X4 = 4	A\$ = "A"
X5 = 5	B\$ = "B"
X6 = 6	C\$ = "C"
X7 = 7	L6\$ = "12"
X8 = 8	N1 = 3
Y5 = 5	K5\$ = "X Y Z"

INSTRUKCJE STERUJĄCE

Instrukcje sterujące służą do zmiany naturalnej kolejności wykonywania programu tj. wykonywania wierszy podług wzrastających numerów.

Instrukcja zatrzymania STOP służy do zakończenia wykonywania programu. Instrukcja składa się ze słowa STOP. Działanie jej w czasie wykonywania programu jest podobne jak działanie wiersza końca programu - wykonywanie programu zostaje zakończone, z odpowiednim wydrukiem, poczem system przechodzi na nasłuch w trybie programowym.

Instrukcja skoku GOTO ma następującą budowę:

$\langle \text{instrukcja skoku} \rangle ::= \text{GOTO } \langle \text{numer wiersza} \rangle$

Po wykonaniu wiersza z instrukcją skoku system nie przechodzi do wykonywania wiersza o następnym wyższym numerze, a do wiersza o numerze podanym w instrukcji skoku, bądź następnego kolejnego, jeśli ten jest wierszem niewykonywanym.

Instrukcja skoku ze śladem GOSUB ma poniższą składnię:

$\langle \text{instr. skoku ze śladem} \rangle ::= \text{GOSUB } \langle \text{numer wiersza} \rangle$

Wykonanie wiersza programu, zawierającego taką instrukcję polega na tym, że system zapamiętuje na specjalnym stosie numer wiersza następnego w kolejności za wierszem z instrukcją GOSUB (czyli zapamiętuje ślad do późniejszego powrotu), poczem przechodzi do wykonywania wiersza programu o numerze podanym w instrukcji skoku ze śladem, bądź do następnego kolejnego wiersza, jeśli ten jest wierszem niewykonywanym.

Instrukcja powrotu według śladu RETURN, składająca się ze słowa RETURN, powoduje przejście do wykonywania programu według ostatnio zapamiętanego na stosie instrukcją GOSUB śladu oraz usunięcie tego śladu ze stosu (a tym samym "odsłonięcie" śladu poprzednio zapamiętanego).

Jeżeli stos nie zawiera już żadnego śladu, wykonywanie programu zostanie wstrzymane z wydrukiem odpowiedniego komunikatu o błędzie. Pojemność stosu śladów wynosi 20, tzn. skoki ze śladem mogą być w czasie wykonywania programu pogrążone w sobie na głębokość 20.

Mechanizm skoku ze śladem i powrotu według śladu umożliwia uzyskanie oszczędności pamięci przez umieszczenie w programie raz tylko kilkakrotnie wykonywanych takich samych fragmentów programu. Poniżej pokazano schematycznie program i kolejność wykonywania (pominięto wiersze nie mające znaczenia dla zmiany sterowania).

Program	100	GOSUB 1000
	110	
	:	
	250	GOSUB 1500
	260	
	:	
	300	GOSUB 1000
	305	
	:	
	600	STOP
	1000	
	:	
	1090	GOSUB 1500
	1100	
	:	
	1300	RETURN
	1500	
	:	
	1630	RETURN
	1800	END

Kolejność wykonywania wierszy w powyższym programie będzie następująca:

wykonany wiersz	zawartość stosu śladów
100	-
1000	110
:	
1090	110, 1100
1500	110, 1100
:	
1680	110
1100	110
:	
1300	-
110	-
:	
250	260
1500	260
:	
1680	-
260	-
:	
300	305
1000	305
:	
1090	305, 1100
1500	305, 1100
:	
1680	305
1100	305
:	
1300	-
305	-
:	
600	-

Instrukcja skoku liczonego ON pozwala na realizację mechanizmu przełącznika - przekazanie sterowania w różne miejsca programu zależnie od wartości wyrażenia arytmetycznego. Budowa tej instrukcji jest następująca:

$\langle \text{instrukcja skoku liczonego} \rangle ::=$

$\text{ON } \langle \text{wyr. wskaźnikowe} \rangle \left\{ \begin{array}{l} \text{GOTO} \\ \text{GOSUB} \end{array} \right\} \langle \text{nr wiersza} \rangle [, \langle \text{nr wiersza} \rangle] \dots$

Przy wykonywaniu takiej instrukcji obliczana jest przez system wartość wyrażenia wskaźnikowego. Jeżeli wynosi ona 1, następuje przejście do wykonywania wiersza programu o numerze równym pierwszemu numerowi z listy za słowem GOTO lub GOSUB, jeżeli 2 - do wiersza o numerze równym drugiemu numerowi z listy itd. Wartości wyrażenia wskaźnikowego mniejsze od jedności lub większe od liczby numerów na liście są błędem i powodują zatrzymanie wykonywania programu, z odpowiednim komunikatem.

W zależności od tego, czy użyto w instrukcji słowa GOTO czy GOSUB wykonywany jest odpowiednio skok bezwarunkowy lub skok ze śladem.

Instrukcja warunkowa IF jest najbardziej w języku BASIC-EXT rozbudowaną instrukcją. Pozwala na uzależnienie sposobu wykonywania programu od spełnienia jednej lub kilku relacji. Budowę i działanie tej instrukcji rozpatrzymy na podstawie kilku jej możliwych postaci. Najpierw podamy definicje pomocniczych elementów takiej instrukcji.

$\langle \text{relacja} \rangle ::= \left\{ \begin{array}{l} \langle \text{wyr. aryt.} \rangle \langle \text{operator relacji} \rangle \langle \text{wyr. aryt.} \rangle \\ \langle \text{wyr. łańc.} \rangle \langle \text{operator relacji} \rangle \langle \text{wyr. łańc.} \rangle \end{array} \right\}$

Operatory relacji i ich znaczenie są następujące:

$=$ równe
 $<>$ nie równe
 $>$ większe
 $<$ mniejsze

$\geq$ lub $\Rightarrow$ nie mniejsze

$\leq$ lub $\leq$ nie większe

Najprostszą postacią instrukcji warunkowej jest postać:

IF <relacja> {
 <instrukcja zatrzymania>
 <instrukcja skoku>
 <instrukcja skoku ze śladem>
 <instrukcja powrotu według śladu>
 <instrukcja przypisania>

Instrukcja przypisania m u s i zaczynać się słowem LET. Działanie takiej postaci instrukcji warunkowej polega na tym, że system sprawdza prawdziwość relacji. Jeśli relacja jest spełniona - wykonuje się instrukcja po relacji, jeśli nie - system nie wykonuje tej instrukcji i przechodzi do wykonywania następnego kolejnego wiersza programu.

Wyjaśnienia wymaga porównywanie wartości łańcuchowych. Dwie wartości łańcuchowe są sobie równe, jeśli mają taką samą długość i wszystkie ich kolejne znaki są odpowiednio równe. Przy nierównych wartościach łańcuchowych porównuje się je według wartości liczbowej kodów ASCII kolejnych znaków, począwszy od pierwszych. Jeżeli jedna z wartości łańcuchowych jest krótsza, można sobie wyobrazić, że jest uzupełniona znakami o kodzie zero do długości drugiej wartości. W celu określenia relacji porównuje się kolejne, począwszy od pierwszych, kody znaków obu wartości łańcuchowych. Większa jest ta wartość, której kod pierwszego ze znaków różniących się między sobą jest większy. Dla łatwiejszego zrozumienia tych zasad podano niżej kilka stałych łańcuchowych, z których każda następna jest większa od wszystkich poprzednich.

" "

" "

" "

" " "

" Ø"

" ØØ"

" 12357"

" 2"

" A"


```
"ANTONI"
"ANTONI K."
"C"
"CØ"
"C11"
"C2"
"ZZZ"
"[X]"
"\
```

Poniższa definicja ilustruje pełne możliwości budowania instrukcji warunkowej.

$$\langle \text{instrukcja warunkowa} \rangle ::= \left\{ \left\{ \{ \text{IF} \langle \text{rel.} \rangle \} \dots \langle \text{instr. przyp.} \rangle \right\} \dots \left[\text{IF} \langle \text{rel.} \rangle \left\{ \begin{array}{l} \langle \text{instr. zatrzymania} \rangle \\ \langle \text{instrukcja skoku} \rangle \\ \langle \text{instr. sk. ze śladem} \rangle \\ \langle \text{instr. powr. wg śladu} \rangle \end{array} \right\} \right. \right. \\ \left. \left. \left\{ \text{IF} \langle \text{relacja} \rangle \right\} \dots \left\{ \begin{array}{l} \langle \text{instrukcja zatrzymania} \rangle \\ \langle \text{instrukcja skoku} \rangle \\ \langle \text{instrukcja skoku ze śladem} \rangle \\ \langle \text{instrukcja powrotu wg śladu} \rangle \end{array} \right\} \right\} \right\}$$

Instrukcja taka wykonywana jest od lewej do prawej. Jeżeli relacja jest spełniona, system wykonuje następujące po niej instrukcje do następnego symbolu IF lub do końca instrukcji. Jeżeli nie jest spełniona - przechodzi do wykonywania wiersza programu następnego w kolejności za wierszem zawierającym instrukcję warunkową.

Następnie, w podobny sposób badana jest następna relacja itd., aż do końca instrukcji. Instrukcje przypisania muszą zaczynać się symbolem LET. Oto przykłady wierszy programu z instrukcjami warunkowymi i ich działanie przy wykonywaniu programu.

```
130 IF L$>="A" IF L$<"C" GOTO 1200 ' SKOCZ GDY LITERA
```

Skok do wiersza 1200 nastąpi tylko wtedy, gdy pierwszym znakiem wartości L\$ będzie duża litera.

220 IF X < M LET M=X; Z=Z+1

Jeżeli wartość X będzie mniejsza od wartości M, nastąpi nadanie zmiennej M wartości zmiennej X oraz zwiększenie wartości zmiennej Z o jedność. Jeżeli nie, nastąpi przejście do wykonywania następnego wiersza programu.

BLOK PĘTLI

Blok pętli jest mechanizmem, pozwalającym powtarzać cyklicznie fragment programu. Blok pętli składa się z:

- wiersza zawierającego instrukcję otwarcia pętli,
- ciała pętli tj. fragmentu programu, którego wykonywanie będzie cyklicznie powtarzane. Ciało pętli może zawierać wiersze z instrukcjami, jak również inne bloki pętli, komentarze itp.,
- wiersza zawierającego instrukcję zamknięcia pętli.

Instrukcja otwarcia pętli FOR ma następującą postać:

<instr.otwarcia pętli> ::=

FOR <zmienna arytm.prosta> = <w.a.> TO <w.a.> [STEP <w.a.>]

Zmienna arytmetyczna prosta, występująca po symbolu FOR jest zmienną sterującą pętli. Jej wartość zmienia się w kolejnych wykonaniach pętli i może być używana jak również zmieniana w ciele pętli.

Instrukcja zamknięcia pętli NEXT ma następującą postać:

<instr.zamknięcia pętli> ::= NEXT <zmienna arytm.prosta>

Zmienna arytmetyczna prosta w tej instrukcji musi być taka sama jak zmienna sterująca odpowiadającej jej instrukcji otwarcia pętli.

Wykonywanie bloku pętli odbywa się według poniższego algorytmu.

Algorytm wykonywania bloku pętli

1. Obliczanie wartości trzech wyrażeń arytmetycznych występujących w instrukcji otwarcia pętli i zapamiętania ich wewnątrz systemu (tzn. tak, że nie są dostępne dla programu):
w1 : wyrażenie arytmetyczne między znakiem równości a napisem T0
w2 : wyrażenie arytmetyczne między napisem T0 a napisem STEP
w3 : wyrażenie arytmetyczne za napisem STEP (równe jedności, gdy napis STEP nie występuje). Jeżeli $w3 \neq 0$ - wydrukowanie komunikatu o błędzie.
2. Sprawdzenie warunku zakończenia pętli:
jeśli $(w2-w1) \neq w3 < 0$: przejście do kroku 6,
jeśli nie - przejście do kroku 3.
3. Nadanie wartości w1 zmiennej sterującej pętli.
4. Wykonanie wierszy programu stanowiących ciało pętli (zawartych między instrukcją otwarcia pętli a instrukcją zamknięcia pętli).
5. Zwiększenie wartości w1 o w3:
 $w1 = w1 + w3$
i przejście do kroku 2.
6. Wykonanie wiersza programu za blokiem pętli.

Przy pisaniu programu należy przestrzegać poniższych zasad. Nie spełnienie ich system wykrywa jako błędy.

1. Bloki pętli mogą zawierać inne bloki pętli, na dowolną głębokość, jednak pętle nie mogą się przecinać.
2. Nie jest dozwolony skok spoza pętli do wnętrza pętli ani do wiersza zawierającego instrukcję zakończenia pętli.
3. Opuszczenie pętli można dokonać poprzez instrukcję zakończenia pętli. Mimo, że jest to możliwe, nie zaleca się stosowania skoków z wnętrza pętli poza blok pętli.

4. Nie jest dozwolony skok z wnętrza pętli do wiersza zawierającego instrukcję otwarcia tej pętli. Jest natomiast dozwolony skok do wiersza zawierającego instrukcję zamknięcia pętli.

Trzeba sobie również zdawać sprawę z tego, że liczby są w maszynie przedstawione w sposób przybliżony, co może w szczególnych przypadkach powodować to, że pętla wykona się np. o jeden raz mniej lub o jeden raz więcej niż to myślał programista układając program. W przypadkach wątpliwych należy używać liczb całkowitych, mających dokładną reprezentację. Tytułem przykładu podano poniżej fragment programu, znajdujący największy i najmniejszy element macierzy A oraz ich wskaźniki

```
200 LET A1=INF ; A9=-INF
210 FOR I=0 TO 10 ' POCZ. PETLI PO "I"
220 FOR J=0 TO 10 ' POCZ. PETLI PO "J"
230 IF A(I,J)<A1 LET A1=A(I,J) ; I1=I ; J1=J
240 IF A(I,J)>A9 LET A9=A(I,J) ; I9=I ; J9=J
250 NEXT J ' KONIEC PETLI PO "J"
260 NEXT I ' KONIEC PETLI PO "I"
```

Po wykonaniu tego fragmentu programu zmienna A1 ma wartość najmniejszego elementu macierzy A, zaś zmienne I1 i J1 wartości wskaźników tego elementu. Zmienna A9 ma wartość największego elementu macierzy A, a zmienne I9, J9 wartości wskaźników tego elementu.

FUNKCJE DEFINIOWANE

Oprócz funkcji standardowych, wbudowanych w język, programista może sobie sam zdefiniować inne funkcje i używać ich w programie. Nazwy funkcji definiowanych zbudowane są jak poniżej:

$$\langle \text{funkcja definiowana} \rangle ::= \left\{ \begin{array}{l} \langle \text{funkcja definiowana arytmetyczna} \rangle \\ \langle \text{funkcja definiowana łańcuchowa} \rangle \end{array} \right\}$$

$$\langle \text{funkcja def. arytmetyczna} \rangle ::= \text{FN} \langle \text{litera} \rangle$$

$$\langle \text{funkcja def. łańcuchowa} \rangle ::= \text{FN} \langle \text{litera} \rangle \langle \text{znak waluty} \rangle$$

Definicje funkcji mogą być jednowierszowe lub wielowierszowe.
Definicja jednowierszowa ma postać:

$$\langle \text{wiersz definicji funkcji} \rangle ::= \langle \text{nr wiersza} \rangle \text{ DEF } \langle \text{funkcj. def.} \rangle$$

$$(\langle \text{par. form. funkcji} \rangle [, \langle \text{par. form. funkcji} \rangle] \dots) = \langle \text{wyrażenie} \rangle$$

$$[\langle \text{komentarz} \rangle]$$

$$\langle \text{parametr formalny funkcji} \rangle ::= \left\{ \begin{array}{l} \langle \text{zmienna prosta} \rangle \\ \langle \text{wektor formalny} \rangle \\ \langle \text{macierz formalna} \rangle \end{array} \right\}$$

$$\langle \text{wektor formalny} \rangle ::= \langle \text{wektor} \rangle ()$$

$$\langle \text{macierz formalna} \rangle ::= \langle \text{macierz} \rangle (,)$$

Typ wyrażenia po prawej stronie znaku równości musi być zgodny z typem funkcji. Zmienne w nawiasach są parametrami formalnymi funkcji. Wyrażenie opisuje sposób obliczenia wartości funkcji. Mogą w nim występować parametry formalne oraz inne zmienne, wołania funkcji standardowych i innych funkcji definiowanych. Rekursja definicji nie jest dozwolona.

Wołanie funkcji definiowanej, które może wystąpić w dowolnym wyrażeniu odpowiedniego typu ma postać:

$$\langle \text{wołanie funkcji definiowanej} \rangle ::=$$

$$\langle \text{funkcja definiowana} \rangle (\langle \text{argument} \rangle [, \langle \text{argument} \rangle] \dots)$$

$$\langle \text{argument} \rangle ::= \left\{ \begin{array}{l} \langle \text{wyrażenie} \rangle \\ \langle \text{wektor formalny} \rangle \\ \langle \text{macierz formalna} \rangle \end{array} \right\}$$

$$\langle \text{wyrażenie} \rangle ::= \left\{ \begin{array}{l} \langle \text{wyrażenie arytmetyczne} \rangle \\ \langle \text{wyrażenie łańcuchowe} \rangle \end{array} \right\}$$

Liczba, kolejność i typ argumentów (parametrów aktualnych) muszą być zgodne z parametrami formalnymi definicji.

Poniżej zdefiniowana funkcja FNT zaokrągla argument do 3 miejsc dziesiętnych.

```
150 DEF FNT(X)=1E-3*INT(1000*X+0.5)
```

Przykłademwołania tej funkcji może być np.

```
FNT(SIN(D*PI/180)+A3)
```

Parametry formalne są lokalne dla definicji tzn. realizacjawołania nie ma wpływu na wartości zmiennych programu o nazwach takich jak parametry formalne.

Wiersz programu, zawierający definicję jest wierszem niewykonywanym i może być umieszczony w dowolnym miejscu programu.

Znacznie większe możliwości dają definicje wielowierszowe. Definicja wielowierszowa (blok definicji) składa się z:

1. Wiersza nagłówka definicji;
2. Ciała definicji, które jest fragmentem programu, służącym do obliczania wartości funkcji;
3. Wiersza końca definicji.

Wiersz nagłówka definicji ma postać:

```
<wiersz nagłówek definicji> ::=
<nr wiersza> DEF <funkcja definiowana> (<parametr formalny
funkcji> [<parametr formalny funkcji>] ...) [ '<komentarz> ]
```

Parametry formalne są lokalne w bloku definicji tzn. użycie ich w definicji nie ma żadnego wpływu na wartości zmiennych o tych samych nazwach występujące w innych miejscach programu.

Wiersz końca definicji ma postać:

$\langle \text{wiersz końca definicji} \rangle ::= \langle \text{nr wiersza} \rangle \text{ FNEND } [' \langle \text{komentarz} \rangle]$

Wołanie funkcji definiowanej definicją wielowierszową ma taką samą postać jak wołanie funkcji definiowanej definicją jednowierszową i może występować w dowolnym miejscu programu w wyrażeniu odpowiedniego typu, za wyjątkiem bloku definicji własnej funkcji.

W ciele definicji, i tylko w nim, musi wystąpić przynajmniej jeden raz wiersz nadania wartości funkcji o postaci:

$\langle \text{wiersz nadania wartości funkcji} \rangle ::=$

$\langle \text{nr wiersza} \rangle \langle \text{funkcja definiowana} \rangle = \langle \text{wyrażenie} \rangle [' \langle \text{komentarz} \rangle]$

bądź nadanie wartości funkcji instrukcją INPUT lub READ.

Cały blok definicji jest fragmentem programu niewykonanym i może znajdować się w dowolnym miejscu programu. Niedozwolone są żadne skoki do wnętrza definicji ani skoki z wnętrza definicji na zewnątrz. Skok do wiersza nagłówka definicji jest równoważny skokowi do wiersza bezpośrednio za blokiem definicji. Blok definicji wykonywany jest wyłącznie w wyniku wykonania instrukcji programu zawierającej wyrażenie, w którym jest wołanie funkcji definiowanej. Wykonanie ciała definicji tzn. realizacja wołania odbywa się według następujących zasad:

1. Funkcji nadawana jest, zależnie od typu, wartość zero lub wartość łańcucha pustego.
2. Obliczane są wartości argumentów będące wyrażeniami. Wartości te będą użyte tam, gdzie w ciele definicji występują odpowiadające im parametry formalne. Mogą być one również zmieniane w czasie wykonywania bloku definicji, co nie ma wpływu na zmienne poza blokiem definicji o takich samych nazwach jak parametry formalne lub parametry aktualne będące zmiennymi prostymi.

3. Tworzone jest połączenie między argumentami będącymi zmiennymi indeksowanymi formalnymi a odpowiednimi parametrami formalnymi. Wszędzie tam w ciele definicji gdzie występują te zmienne indeksowane, operacja będzie dotyczyła odpowiedniej zmiennej indeksowanej w programie. Zmienne indeksowane są zatem wołane przez nazwę, a nie przez wartość.
4. Wykonanie wiersza nadania wartości funkcji powoduje obliczenie wartości wyrażenia po prawej stronie znaku równości i nadanie jego wartości funkcji.
5. W ciele definicji może w wyrażeniach występować samodzielnie nazwa danej funkcji definiowanej (bez listy parametrów). Reprezentuje ona w czasie obliczania funkcji jej aktualną wartość.
6. W ciele definicji mogą występować również inne zmienne programu. Definicja może zatem mieć efekty uboczne, na co trzeba zwrócić uwagę przy pisaniu programu.

Zasady wyżej opisane zilustrujemy przykładami. Poniżej podana jest definicja funkcji obliczającej wartość wielomianu

$$a_0 + a_1x + a_2x^2 + \dots + a_nx^n$$

oraz fragment programu używający tej funkcji do obliczenia wartości funkcji e^t drogą sumowania $k+1$ wyrazów szeregu

$$e^t = 1 + t + \frac{1}{2!} t^2 + \frac{1}{3!} t^3 + \dots$$

```
100 DEF FNW(X,A(),N)
110 FOR N=N TO 0 STEP -1
120 FNW=FNW*X+A(N)
130 NEXT N
140 FEND
```

```
500 C(0)=1 ' OBL. WEKTORA WSPOLCZ.
510 FOR I=1 TO K
520 C(I)=C(I-1)/I
530 NEXT I ' KONIEC OBL. WSPOLCZ.
540 F=FNW(T,C(),K)
```


Po wykonaniu tego fragmentu programu zmienna F będzie miała wartość przybliżenia e^t .

Wartością funkcji, której definicję podano nizej jest ten spośród N elementów wektora łańcuchowego będącego argumentem, który zawiera łańcuch najniższy w porządku alfabetycznym.

```
200 DEF FNA$(X$( ),N)
210 FNA$=CHR$(127) ' INICJALIZ. WART. FUNKCJI
220 FOR N=1 TO N
230 IF X$(N)>FNA$ GOTO 250
240 FNA$=X$(N)
250 NEXT N
260 FNEED
```

Wartością nizej zdefiniowanej funkcji FNX jest większy z jej dwóch argumentów

```
300 DEF FNX(A,B)
310 FNX=A
320 IF B<=A GOTO 340
330 FNX=B
340 FNEED
```

Chcąc np. użyć w programie, w wyrażeniu arytmetycznym największej z wartości zmiennych X1, X2, X3, X4 możemy to uzyskać za pomocą poniższegowołania funkcji FNX

FNX(FNX(X1,X2),FNX(X3,X4))

Funkcja, której definicję podano na następnej stronie jest trochę bardziej skomplikowana. Została ona skonstruowana z myślą o przetwarzaniu tekstów dla celów wydawniczych. Większość używanych końcówek nie daje możliwości wprowadzania z klawiatury małych liter. Funkcja pozwala w przygotowanym odpowiednio fragmencie tekstu zamienić duże litery na małe. Wartością funkcji jest wartość argumentu, który został przetworzony kolejno, znak po znaku według poniższych reguł:


```
100 DEFFNL$(A$)
110 FOR Q1=1 TO LEN(A$)
120 LET Q2=COD(Q1,A$)
130 IF Q2=32 GOTO 260 ' IF NOT CNTRL CHAR.
140 IF Q2=27 GOTO 280 ' IF "ESCAPE" CHAR.
150 IF Q2<>14 GOTO 500
160 REM - AFTER "SHIFT OUT" CHAR.
170 LET Q1=Q1+1
180 IF Q1>LEN(A$) GOTO 500
190 LET Q2=COD(Q1,A$)
200 IF Q2=15 GOTO 500 ' IF "SHIFT IN" CHAR.
220 IF Q2<32 GOTO 170 ' IF CNTRL CHAR.
230 FNL$=FNL$&CHR$(Q2)
240 IF Q2>64 IF Q2<=90 GOTO 170 ' IF LETTER
250 GOTO 500
260 IF Q2<>46 GOTO 330
270 FNL$=FNL$&CHR$(Q2) ' AFTER PERIOD
280 LET Q1=Q1+1
290 IF Q1>LEN(A$) GOTO 500
300 LET Q2=COD(Q1,A$)
310 IF Q2=32 GOTO 270 ' IF SPACE
320 GOTO 490
330 IF Q2>64 IF Q2<=90 LET Q2=Q2+32 ' CHANGE INTO SMALL IF LETTER
490 FNL$=FNL$&CHR$(Q2)
500 NEXT Q1
510 FNEED
900 REM
920 REM
940 REM
1000 REM *** MAIN PROGRAM ***
1002 REM
1005 LET F1=1
1010 PRINT "INPUT FILE NAME ? ";
1020 INPUT F$
1030 FILE #F1: F$
1035 IF PER(F1,"LINPUT")=1 GOTO 1065
1040 PRINT "WRONG INPUT FILE NAME"
1050 LET F1=F1+1
1060 GOTO 1010
1065 LET F2=F1+1
1070 PRINT "OUTPUT FILE NAME ? ";
1080 INPUT F$
1090 FILE #F2: F$
1100 IF PER(F2,"PRINT")=1 GOTO 1140
1110 PRINT "WRONG OUTPUT FILE NAME"
1120 LET F2=F2+1
1130 GOTO 1070
1140 RESET #F2:
1145 REM HERE FOLLOWS THE PORTION THAT DOES THE REAL WORK
1150 IF PER(F1,"LINPUT")=0 STOP ' STOP IF INPUT FILE EXHAUSTED
1160 LINPUT #F1: L$
1170 PRINT #2: FNL$(L$)
1180 GOTO 1150
1200 END ' OF THE WHOLE PROGRAM
*
```


1. Duże litery zostają zamienione na małe litery z poniższymi wyjątkami:
 - a) jeżeli w łańcuchu występuje znak ESC i bezpośrednio po nim litera, to litera ta nie jest zamieniana na małą. Znak ESC jest usuwany;
 - b) jeżeli wystąpił znak SO, to występujące po nim litery nie są zamieniane na małe, aż do wystąpienia znaku SI bądź znaku różnego od litery. Znaki SO i SI nie są przenoszone do wartości funkcji;
 - c) jeżeli w łańcuchu będącym argumentem wystąpiła kropka, to następująca po niej bezpośrednio, lub oddzielona od niej tylko spacjami litera jest pozostawiona jako litera duża.
2. Wszystkie znaki sterujące (tj. znaki o kodach o wartości poniżej 32) zostają usunięte.

W bloku definicji tej funkcji są użyte dwie zmienne nielokalne @1 i @2. Jeżeli w programie, poza tym blokiem definicji, występują zmienne o takich nazwach, to każdewołanie funkcji FNL\$ zmieni ich wartość.

OPERACJE NA ZMIENNYCH INDEKSOWANYCH

Oprócz omówionych już operacji, umożliwiających działania na elementach zmiennych indeksowanych, język BASIC zawiera instrukcje, pozwalające wykonywać działania na zmiennych indeksowanych traktowanych jako całości.

Jak już mówiliśmy, wystąpienie zmiennej indeksowanej w programie powoduje ustalenie zakresów wartości indeksów tak, by mogły one przyjmować wartości od 0 do 10 oraz, dla zmiennych indeksowanych arytmetycznych, zarezerwowanie odpowiedniego obszaru na przechowanie wartości elementów zmiennych indeksowanych. Umożliwia to używanie w programach zmiennych indeksowanych bez żadnej specjalnej deklaracji.

Jeżeli jednak występuje w programie

potrzeba użycia zmiennych indeksowanych o większych rozmiarach (lub mniejszych, celem oszczędności pamięci), można to uzyskać za pomocą deklaracji wymiarów.

Deklaracja wymiarów zmiennych indeksowanych ma następującą postać:

$$\begin{aligned} &\langle \text{deklaracja wymiarów} \rangle ::= \\ \text{DIM} &\left\{ \begin{array}{l} \langle \text{wektor} \rangle (\langle \text{wymiar} \rangle) \\ \langle \text{macierz} \rangle (\langle \text{wymiar} \rangle, \langle \text{wymiar} \rangle) \end{array} \right\} \left[, \left\{ \begin{array}{l} \langle \text{wektor} \rangle (\langle \text{wymiar} \rangle) \\ \langle \text{macierz} \rangle (\langle \text{wymiar} \rangle, \langle \text{wymiar} \rangle) \end{array} \right\} \right] \\ &\langle \text{wymiar} \rangle ::= \langle \text{liczba naturalna} \rangle \end{aligned}$$

Wiersz, zawierający deklarację wymiarów, o postaci:

$$\langle \text{nr wiersza} \rangle \langle \text{deklaracja wymiarów} \rangle [' \langle \text{komentarz} \rangle]$$

może być umieszczony w dowolnym miejscu programu i jest wierszem niewykonanym. Jeden program może zawierać kilka wierszy deklaracji wymiarów, które łącznie mają takie znaczenie, jakby były jedną deklaracją wymiarów. Ta sama zmienna nie może mieć dwukrotnie deklarowanych wymiarów - system wykryje to jako błąd, przy uruchomieniu programu.

Deklaracja wymiarów powoduje zarezerwowanie pamięci na przechowywanie elementów zmiennych indeksowanych w taki sposób, by wskaźniki tych zmiennych mogły przyjmować wartości całkowite od zera do wartości podanych w deklaracji wymiarów. Jeżeli zatem w programie jest wiersz

$$50 \text{ DIM } V(3), A\$(2,3)$$

to system zarezerwuje miejsce na przechowanie elementów wektora V:

$$V(0), V(1), V(2), V(3)$$

i elementów macierzy A\$

A\$(0,0)	A\$(0,1)	A\$(0,2)	A\$(0,3)
A\$(1,0)	A\$(1,1)	A\$(1,2)	A\$(1,3)
A\$(2,0)	A\$(2,1)	A\$(2,2)	A\$(2,3)

Wskaźniki zmiennych indeksowanych występujących w programie, a nie występujących w deklaracji wymiarów, mogą jak już uprzednio powiedziano, przyjmować wartości od zera do 10. System musi jednak mieć możliwość rozpoznania, czy ma do czynienia z macierzą, czy z wektorem.

Rezerwacja pamięci na przechowywanie zmiennych indeksowanych arytmetycznych ma zatem charakter statyczny i jest dokonywana w momencie uruchomienia programu. Ze zmiennymi indeksowanymi łańcuchowymi rzecz ma się nieco inaczej, ponieważ długość ich elementów może się w czasie wykonywania programu zmieniać. Rezerwacja pamięci dokonywana w wyniku deklaracji wymiarów zmiennych indeksowanych łańcuchowych dotyczy zatem tylko tablic, zawierających informacje o rozmieszczeniu elementów takiej zmiennej.

Przy wykonywaniu operacji na całych zmiennych indeksowanych, w czasie wykonywania programu, możemy mieć do czynienia jeszcze z wymiarami aktualnymi, które mogą się zmieniać w czasie wykonywania programu, tak jednak tylko, aby liczba elementów zmiennej indeksowanej nie przekroczyła liczby wynikającej z deklaracji wymiarów danej zmiennej indeksowanej. Przewymiarowanie takie może być jawne lub niejawne. Wyjaśni to się w dalszej części wywodu.

Do wykonywania operacji na zmiennych indeksowanych jako całości służy instrukcja przypisania zmiennych indeksowanych o budowie podanej na następnej stronie.

Instrukcja ta zaczyna się zawsze słowem MAT. Obiekt po lewej stronie znaku równości będzie po wykonaniu instrukcji zawierał wynik operacji. Napisy po prawej stronie znaku równości określają rodzaj operacji i argumenty. Wymiary aktualne argumentów muszą być zgodne z wymogami rachunku macierzy. Macierz lub wektor wynikowy mogą ulec w wyniku wykonania instrukcji przewymiarowaniu - ich wymiary aktualne po wykonaniu instrukcji będą zawsze takie, jak wymiary wyniku operacji. Operacje NUL\$, ZER, CON, IDN pozwalają na jawne przewymiarowanie obiektu wynikowego - nowe wymiary aktualne będą równe wartości wyrażeń wskaźnikowych występujących

$\langle \text{instrukcja przypisania zmiennych indeksowanych} \rangle ::=$

$$\text{MAT } \langle \text{wektor łańc.} \rangle = \left\{ \begin{array}{l} \langle \text{wektor łańcuchowy} \rangle \\ \text{NUL}\$ \left[\left(\langle \text{wyrażenie wskaźnikowe} \rangle \right) \right] \end{array} \right\}$$

$$\text{MAT } \langle \text{macierz łańc.} \rangle = \left\{ \begin{array}{l} \langle \text{macierz łańcuchowa} \rangle \\ \text{NUL}\$ \left[\left(\langle \text{wyr. wskaźn.} \rangle, \langle \text{wyr. wskaźn.} \rangle \right) \right] \end{array} \right\}$$

$$\text{MAT } \langle \text{wektor arytm.} \rangle = \left\{ \begin{array}{l} \langle \text{wektor arytmetyczny} \rangle \\ \langle \text{wektor arytm.} \rangle \left\{ \begin{array}{c} + \\ - \end{array} \right\} \langle \text{wektor arytm.} \rangle \\ \langle \text{wyr. arytm.} \rangle \# \langle \text{wektor arytm.} \rangle \\ \langle \text{wektor arytm.} \rangle \# \langle \text{macierz arytm.} \rangle \\ \langle \text{macierz arytm.} \rangle \# \langle \text{wektor arytm.} \rangle \\ \left\{ \begin{array}{c} \text{ZER} \\ \text{CON} \end{array} \right\} \left[\left(\langle \text{wyrażenie wskaźnikowe} \rangle \right) \right] \end{array} \right\}$$

$$\text{MAT } \langle \text{macierz arytm.} \rangle = \left\{ \begin{array}{l} \langle \text{macierz arytmetyczna} \rangle \\ \langle \text{macierz arytm.} \rangle \left\{ \begin{array}{c} + \\ - \end{array} \right\} \langle \text{macierz arytm.} \rangle \\ \langle \text{wyr. arytm.} \rangle \# \langle \text{macierz arytm.} \rangle \\ \langle \text{macierz arytm.} \rangle \# \langle \text{macierz arytm.} \rangle \\ \left\{ \begin{array}{c} \text{ZER} \\ \text{CON} \\ \text{IDN} \end{array} \right\} \left[\left(\langle \text{wyr. wskaźn.} \rangle, \langle \text{wyr. wskaźn.} \rangle \right) \right] \\ \left\{ \begin{array}{c} \text{TRN} \\ \text{INV} \end{array} \right\} \langle \text{macierz arytmetyczna} \rangle \end{array} \right\}$$

$$\text{MAT } \langle \text{zmienna arytm.} \rangle = \langle \text{wektor arytm.} \rangle \# \langle \text{wektor arytm.} \rangle$$

w nawiasach za operacją. Operacja NUL δ daje zmienną indeksowaną, której wszystkie elementy są łańcuchami pustymi. Operacja ZER daje zmienną indeksowaną, której wszystkie elementy mają wartość zero. Operacja CON daje zmienną indeksowaną, której wszystkie elementy są równe jedności. Operacja IDN daje macierz jednostkową - macierz kwadratową, której elementy leżące na głównej przekątnej mają wartość równą jedności, a wszystkie pozostałe elementy wartość zero.

Instrukcja przypisania zmiennych indeksowanych arytmetycznych nie dotyczy elementów o wskaźnikach równych zeru. Elementy te są jednak używane przez system na miejsca robocze i ich wartości po wykonaniu takiej instrukcji są nieokreślone.

Operacja TRN dokonuje transpozycji macierzy. Operacja INV dokonuje odwrócenia macierzy. Nie jest sygnalizowany błąd, gdy macierz będąca argumentem tej operacji jest osobliwa. Można się o tym dowiedzieć w programie, badając wartość funkcji standardowej DET, będącą wartością wyznacznika ostatnio odwróconej macierzy.

Wartością zmiennej standardowej ROW jest aktualny pierwszy wymiar (liczba wierszy) zmiennej indeksowanej, która była wynikiem ostatnio wykonanej instrukcji przypisania zmiennych indeksowanych. Podobnie, wartością zmiennej standardowej COL jest aktualny drugi wymiar (liczba kolumn), jeśli był wektor - jedność. Instrukcja postaci:

MAT A = A

lub

MAT B δ = B δ

pozwała uzyskać te wartości, bez żadnych dodatkowych operacji.

Z opisu instrukcji przypisania zmiennych indeksowanych wynika, że można wykonywać następujące operacje:

- kopiowanie zmiennych indeksowanych
- dodawanie i odejmowanie wektorów i macierzy

- mnożenie wektorów i macierzy przez skalar (wyrażenie arytmetyczne)
- mnożenie wektora przez macierz i macierzy przez wektor
- mnożenie macierzy przez macierz; macierz wynikowa nie może tu być ta sama co jeden z argumentów
- generacja stałych macierzowych
- transpozycja macierzy
- odwracanie macierzy
- mnożenie wektora przez wektor (iloczyn skalarny).

OPERACJE WEJŚCIA/WYJŚCIA DLA INFORMACJI ZNAKOWEJ

Język BASIC jest wyposażony w instrukcje, pozwalające programom wprowadzać i wyprowadzać informację w postaci ciągów znaków ASCII (ISO-7). Źródłem wprowadzanej informacji znakowej lub przeznaczeniem wyprowadzanej mogą być:

- końcówka użytkownika
- urządzenia peryferyjne, np. czytnik taśmy papierowej, drukarka
- zbiory znakowe.

Można sobie wyobrazić, że informacja jest wprowadzana i wyprowadzana poprzez strumienie, które będziemy nazywali kanalami. Kanały w programach BASIC są rozróżniane poprzez liczby całkowite nieujemne. Do powiązania kanału z odpowiednim zbiorem czy urządzeniem służy instrukcja otwarcia kanału.

<instrukcja otwarcia kanału> ::=

FILE # <kanał> : <wyrażenie łańcuchowe>

<kanał> ::= <wyrażenie wskaźnikowe>

Wyrażenie wskaźnikowe po znaku numeru jest traktowane jako numer kanału. Wartość wyrażenia łańcuchowego po dwukropku musi mieć jedną z dwóch postaci:

$$\left\{ \begin{array}{l} \langle \text{nazwa zbioru} \rangle \\ \langle \text{urządzenie wspólne} \rangle \end{array} \right\}$$

$\langle \text{nazwa zbioru} \rangle ::= \langle \text{identyfikator} \rangle$

$\langle \text{identyfikator} \rangle ::= \langle \text{litera} \rangle \left[\begin{array}{l} \langle \text{litera} \rangle \\ \langle \text{cyfra} \rangle \end{array} \right] \dots$

System uwzględnia tylko 6 pierwszych znaków identyfikatora, dalsze są ignorowane.

$\langle \text{urządzenie wspólne} \rangle ::= \langle \text{nazwa urządzenia} \rangle \text{xxx}$

$\langle \text{nazwa urządzenia} \rangle ::= \langle \text{litera} \rangle \left\{ \begin{array}{l} \langle \text{litera} \rangle \\ \langle \text{cyfra} \rangle \end{array} \right\} \left[\begin{array}{l} \langle \text{litera} \rangle \\ \langle \text{cyfra} \rangle \end{array} \right]$

Urządzenia wspólne są zdefiniowane a priori.

System BASIC dostarczany użytkownikowi ma zdefiniowane następujące urządzenia wspólne:

TR1xxx czytnik taśmy papierowej
 PUNxxx dziurkarka taśmy papierowej
 LP2xxx drukarka
 MO1xxx końcówka użytkownika jako urządzenie wejściowe
 MO0xxx końcówka użytkownika jako urządzenie wyjściowe

Działanie instrukcji otwarcia kanału jest następujące:

1. Po obliczeniu wartości wyrażenia wskaźnikowego system sprawdza, czy kanał taki nie jest już otwarty. Jeśli tak - program zostaje zatrzymany z wydrukiem o błędzie.
2. Obliczana jest i sprawdzana poprawność wyrażenia łańcuchowego. Jeżeli wyrażenie jest poprawne, otwierany jest dostęp do danego zbioru czy urządzenia

i tworzone połączenie, umożliwiające ~~wszelkie~~ późniejsze odwołania poprzez numer kanału.

W przypadku zbioru, przeszukiwane są skorowidze i po znalezieniu zbioru otwierany jest do niego dostęp. Jeżeli zbioru o podanej nazwie nie ma w przeszukiwanych skorowidzach, system BASIC tworzy nowy zbiór o danej nazwie i otwiera do niego dostęp. Nie jest w tym momencie sprawdzany typ zbioru (określony przez zawartość) ani to, czy wprowadzanie bądź wyprowadzanie jest dozwolone, gdyż nie wiadomo jeszcze, jakie operacje we/wy będą na danym kanale wykonywane, tzn. czy kanał będzie kanałem wejściowym czy wyjściowym.

Nowo otwarty instrukcją otwarcia kanału zbiór jest zbiorem pustym i nie ma określonego typu. Typ zbioru jest zawsze określony przez jego zawartość. Znakowe operacje we/wy można wykonywać na zbiorach typu CHR. Jeżeli zbiór (a nie kanał!) jest już otwarty, to stan jego nie ulega żadnej zmianie, a tworzone jest tylko połączenie umożliwiające dostęp do tego zbioru poprzez otwierany właśnie kanał.

Kilka kanałów może więc dawać jednocześnie dostęp do jednego zbioru lub urządzenia, natomiast jeden kanał nie może być połączony z więcej niż jednym zbiorem lub urządzeniem.

Z kanałem współpracującym ze zbiorem znakowym związane są:

- typ zbioru (w tym przypadku znakowy)
- długość zbioru
- margines (dla kanału wyjściowego)
- wskaźnik zbioru.

Margines określa liczbę znaków w wierszu dla kanału wyjściowego znakowego. Liczby tej używa system BASIC przy określaniu miejsca, w którym trzeba przejść do nowego wiersza, by nie przekroczyć szerokości wiersza. Wartość

marginisu można zmienić instrukcją ustawiania marginisu o postaci:

$\langle \text{instrukcja ustawiania marginisu} \rangle ::=$
 $\text{MARGIN}[\# \langle \text{kanał} \rangle :] \langle \text{wyrażenie wskaźnikowe} \rangle$

Wykonanie tej instrukcji powoduje ustalenie dla danego kanału marginisu o wartości wynikającej z wartości wyrażenia wskaźnikowego. Margines kanału wyjściowego związanego ze zbiorem znakowym lub urządzeniem można zmienić w każdym momencie wykonywania programu.

Pominięcie wskaźnika kanału w tej instrukcji tj. użycie instrukcji o postaci

$\text{MARGIN} \langle \text{wyrażenie wskaźnikowe} \rangle$

spowoduje, że system potraktuje ją jako odnoszącą się do kanału zerowego. Kanał zerowy jest zawsze otwarty i przypisany do końcówki użytkownika. Podobnie traktowane jest pominięcie wyróżnika kanału w innych instrukcjach (tam, gdzie jest to dozwolone).

Wskaźnik zbioru znakowego jest przechowywany wewnątrz systemu BASIC i służy do oznaczania miejsca, którego będzie dotyczyła następna instrukcja we/wy. Informacje znakowe są zawsze przesyłane sekwencyjnie. Po wykonaniu instrukcji otwarcia kanału wskaźnik będzie ustawiony na początek zbioru. Wprowadzanie zatem nastąpi od początku zbioru, wyprowadzanie zaś informacji znakowej spowoduje zniszczenie informacji zapisanej uprzednio w zbiorze. Wskaźnik zbioru dla kanału znakowego można w dowolnym momencie przesunąć na początek zbioru instrukcją o postaci:

$\text{RESET}[\# \langle \text{kanał} \rangle :]$

Po wykonaniu tej instrukcji zbiór nie jest ani wejściowym ani wyjściowym. O tym, jaki będzie, zadecyduje wykonana operacja we/wy.

Instrukcja wprowadzania ma następującą postać:

$$\langle \text{instrukcja wprowadzania} \rangle ::= \\ \text{INPUT} \left[\# \langle \text{kanał} \rangle : \left\{ \begin{array}{l} \langle \text{zmienna} \rangle \\ \langle \text{fun. defin.} \rangle \end{array} \right\} \left[, \left\{ \begin{array}{l} \langle \text{zmienna} \rangle \\ \langle \text{fun. defin.} \rangle \end{array} \right\} \right] \dots \right]$$

Działanie instrukcji jest takie, że pobiera z wejścia dane w postaci znakowej i przypisuje odpowiadające im wartości kolejnym zmiennym lub funkcjom definiowanym z listy instrukcji wprowadzania. Funkcje definiowane mogą wystąpić tylko wewnątrz bloku swojej definicji. Typy danych i zmiennych muszą być odpowiednio zgodne. Poszczególne dane są rozdzielone przecinkami lub nowymi wierszami. W przypadku danych liczbowych można jako separatora danych używać również średnika albo dwóch lub więcej spacji. Dane łańcuchowe mogą być zapisane w postaci łańcuchów otwartych lub zamkniętych w znaki cudzysłowu.

Pominięcie w instrukcji wyróżnika kanału powoduje, że dane są pobierane z końcówki użytkownika i instrukcja dodatkowo:

- wyprowadza na końcówkę przed pobraniem danych znak zapytania, chyba że przedtem wyprowadzono na końcówkę tekst instrukcją drukowania zakończoną średnikiem,
- wyprowadza na końcówkę znak zapytania, jeśli wprowadzony z końcówki wiersz danych nie zawierał liczby danych wystarczającej do wyczerpania listy zmiennych.

Jeżeli, w momencie gdy program oczekuje na wprowadzenie danych z końcówki, podamy z niej znaki DC4 i LF, spowodujemy zatrzymanie programu i przejście na nasłuch w trybie programowym.

Natrafienie na koniec zbioru przed wyczerpaniem listy zmiennych powoduje zatrzymanie programu z wydrukiem o błędzie.

Instrukcja wprowadzania wierszami służy do wprowadzania wartości łańcuchowych, zawierających znaki, nie dające się wprowadzić zwykłą instrukcją wprowadzania, np. przecinki, cudzysłowy. Ma ona następującą postać:

$$\langle \text{instrukcja wprowadzania wierszami} \rangle ::= \\ \text{LINPUT} \left[\# \langle \text{kanał} \rangle : \right] \langle \text{zmienna łańcuch.} \rangle \left[, \langle \text{zmienna łańcuch.} \rangle \right] \dots$$

Wykonanie tej instrukcji powoduje pobieranie znaków z wejścia aż do znaku końca wiersza LF, utworzenie z nich wartości łańcuchowej (bez znaku LF) i nadanie jej pierwszej zmiennej z listy, pobranie następnego wiersza, utworzenie z niego wartości łańcuchowej i nadanie jej wartości drugiej zmiennej z listy itd. aż do wyczerpania listy zmiennych. Poza tym działanie tej instrukcji jest podobne do działania instrukcji wprowadzania INPUT.

Instrukcja wprowadzania zmiennych indeksowanych służy do wprowadzania całych zmiennych indeksowanych - wektorów i macierzy. Jej postać jest następująca:

$$\langle \text{instrukcja wprowadzania zmiennych indeksowanych} \rangle ::= \\ \text{MAT INPUT} \left[\# \langle \text{kanał} \rangle : \right] \left\{ \begin{array}{l} \langle \text{wektor} \rangle \left[\left(\langle \text{wyr. wskaźn.} \rangle \right) \right] \\ \langle \text{macierz} \rangle \left[\left(\langle \text{w.w.} \rangle , \langle \text{w.w.} \rangle \right) \right] \end{array} \right\} \\ \left[, \left\{ \begin{array}{l} \langle \text{wektor} \rangle \left[\left(\langle \text{wyr. wsk.} \rangle \right) \right] \\ \langle \text{macierz} \rangle \left[\left(\langle \text{wyr. wskazn.} \rangle , \langle \text{wyr. wskazn.} \rangle \right) \right] \end{array} \right\} \right] \dots$$

Instrukcja, podobnie jak wszystkie inne instrukcje wykonujące operacje na całych zmiennych indeksowanych arytmetycznych, nie dotyczy elementów o wskaźnikach zerowych, zaś dla zmiennych indeksowanych łańcuchowych dotyczy również elementów o wskaźnikach zero.

Można tą instrukcją, jak widać z jej budowy, dokonać jawnego przewymiarowania zmiennych indeksowanych. W czasie

wykonywania instrukcja pobiera z wejścia taką liczbę danych, jaka wynika z aktualnych wymiarów lub z przewymiarowania zmiennej indeksowanej z listy i nadaje wartości tych danych kolejnym elementom. Macierze wprowadzane są wierszami. Np. wiersz programu

```
300 MAT INPUT X(2,3)
```

spowoduje wprowadzenie z końcówki 6 liczb i nadanie ich wartości kolejno elementom $X(1,1)$, $X(1,2)$, $X(1,3)$, $X(2,1)$, $X(2,2)$, $X(2,3)$. Poza tym instrukcja działa podobnie jak instrukcja wprowadzania.

Instrukcja wprowadzania wierszami zmiennych indeksowanych,
o postaci:

$$\langle \text{instr. wprowadzania wierszami zmiennych indeks.} \rangle ::=$$
$$\text{MAT LINPUT} \left[\# \langle \text{kanał} \rangle : \left\{ \begin{array}{l} \langle \text{wektor łańc.} \rangle [(\langle \text{wyr. wskaźn.} \rangle)] \\ \langle \text{macierz łańc.} \rangle [(\langle \text{w.w.} \rangle, \langle \text{w.w.} \rangle)] \end{array} \right\} \right]$$
$$\left[\left\{ \begin{array}{l} \langle \text{wektor łańc.} \rangle [(\langle \text{wyr. wskaźn.} \rangle)] \\ \langle \text{macierz łańc.} \rangle [(\langle \text{w.w.} \rangle, \langle \text{w.w.} \rangle)] \end{array} \right\} \right] \dots$$

pozwała na wprowadzanie, w podobny sposób jak poprzednio opisana, całych zmiennych indeksowanych łańcuchowych. Z każdego wprowadzonego wiersza zostanie utworzona wartość łańcuchowa i nadana kolejnemu elementowi zmiennej indeksowanej.

Poniższy fragment programu

```
100 DIM P$(100)
110 INPUT N
120 IF N<1 STOP
130 IF N>100 STOP
140 MAT INPUT P$(N)
```


będzie miał następujące działanie:

1. Zostanie wyprowadzony na końcówkę znak zapytania
2. Z końcówki zostanie wprowadzona podana przez użytkownika liczba i jej wartość nadana zmiennej N
3. Jeżeli N jest mniejsze od 1 lub większe od 100, wykonywanie programu zostanie zatrzymane
4. N+1 razy zostanie:
 - a/ wyprowadzony na końcówkę znak zapytania
 - b/ pobrany z końcówki wiersz informacji
 - c/ utworzona z tego wiersza wartość łańcuchowa zostanie nadana kolejnemu, począwszy od zerowego, elementowi wektora łańcuchowego P\$.

Po wykonaniu tego fragmentu programu aktualny wymiar wektora będzie wynosił N. Próby odwołania się do elementów tego wektora o wskaźnikach większych od N zostaną potraktowane jako błąd.

Dokładnie równoważne do powyższego działanie będzie miał poniższy fragment programu:

```
100 DIM P$(100)
110 INPUT N
120 IF N<1 STOP
130 IF N>100 STOP
140 MAT P$=NUL$(N)
150 FOR I=0 TO N
160 LINPUT P$(I)
170 NEXT I
```

Wiersz 140 w powyższym programie, pozornie niepotrzebny, służy do ustalenia aktualnego wymiaru wektora P\$. Wiersze 150, 160, 170 można, bez zmiany działania programu zastąpić jednym wierszem

```
150 MAT LINPUT P$
```

Nie potrzeba tu przewymiarowania w instrukcji LINPUT, gdyż dokonaliśmy go już w wierszu 140.

Instrukcja drukowania jest podstawową instrukcją służącą do wyprowadzania informacji w postaci znakowej na końcówkę, na urządzenia wspólne lub do zbiorów znakowych. Pełna jej składnia przedstawia się następująco:

$$\begin{aligned} &\langle \text{instrukcja drukowania} \rangle ::= \text{PRINT} [\# \langle \text{kanał} \rangle :] \\ &\left\{ \begin{array}{l} [,] \dots \left[\left\{ \begin{array}{l} \langle \text{wyrażenie} \rangle \\ \langle \text{tabulator} \rangle \end{array} \right\} \left[\left\{ \begin{array}{l} [,] \dots \\ ; \end{array} \right\} \left\{ \begin{array}{l} \langle \text{wyrażenie} \rangle \\ \langle \text{tabulator} \rangle \end{array} \right\} \dots \left[\begin{array}{l} [,] \dots \\ ; \end{array} \right] \right] \right] \\ \text{USING} \left\{ \begin{array}{l} \langle \text{nr wiersza} \rangle \\ \langle \text{wyr. łańc.} \rangle \end{array} \right\} : \langle \text{wyrażenie} \rangle [, \langle \text{wyrażenie} \rangle] \dots [;] \end{array} \right\} \\ &\langle \text{tabulator} \rangle ::= \text{TAB} (\langle \text{wyrażenie wskaźnikowe} \rangle) \end{aligned}$$

Najprostsza postać tej instrukcji

$\text{PRINT} [\# \langle \text{kanał} \rangle :]$

powoduje w czasie wykonywania wyprowadzenie znaków końca wiersza, tzn. znaków CR i LF. Pominięcie wyróżnika kanału w instrukcji drukowania, podobnie jak i we wszystkich instrukcjach wprowadzania, powoduje wyprowadzenie informacji na końcówkę. Postać bez słowa USING tzn. opisana górnym wierszem w wyżej podanej definicji służy do drukowania swobodnego. Kolejne elementy takiej instrukcji, znajdujące się za wyróżnikiem kanału lub za słowem PRINT, jeśli wyróżnika kanału nie ma, są interpretowane w czasie wykonywania w niżej podany sposób:

a/ wyrażenie arytmetyczne

Obliczana jest wartość wyrażenia, poczem następuje wyprowadzenie znaków reprezentacji dziesiętnej tej wartości w następujący sposób:

- jeżeli liczba jest ujemna, wyprowadzany jest znak minus, jeżeli dodatnia - spacja;
- jeżeli liczba jest liczbą całkowitą co najwyżej

sześciocyfrową - wyprowadzane są znaki cyfr tworzących tę liczbę;

- jeżeli liczba da się przedstawić w postaci ułamkowej z kropką dziesiętną o co najwyżej sześciu cyfrach znaczących - jest wyprowadzana w tej postaci. Ostatnia cyfra jest, w razie potrzeby zaokrąglana. Zera nieznaczące nie są wyprowadzane;
- jeżeli liczba nie da się przedstawić w żadnej z powyższych postaci, jest wyprowadzana w postaci półlogarytmicznej - ułamka i całkowitego wykładnika potęgi dziesięciu, przez którą należy ten ułamek pomnożyć.

Najpierw wyprowadzany jest ułamek z jedną cyfrą przed kropką i pięcioma po kropce (ostatnia cyfra zaokrąglona), następnie litera E, znak i wartość wykładnika;

- po liczbie wyprowadzana jest jedna spacja.

b/ wyrażenie łańcuchowe

Obliczana jest wartość wyrażenia, poczem wyprowadzana znak po znaku.

c/ przecinek

Wystąpienie przecinka powoduje to, że następny element będzie drukowany nie od tej pozycji wiersza, gdzie zakończyło się drukowanie poprzedniego elementu, a od pozycji o numerze będącym najbliższą wielokrotnością 15 (pierwszy znak w wierszu ma pozycję numer zero) - tzn. następuje wyprowadzenie odpowiedniej liczby spacji.

d/ średnik

Służy jako separator - nie powoduje wyprowadzenia żadnych znaków.

e/ tabulator

Obliczana jest wartość argumentu tabulatora. Następny element będzie drukowany nie od tej pozycji w wierszu, na której zakończyło się drukowanie poprzedniego elementu, a od pozycji o numerze równym argumentowi tabu-

latora. Jeżeli argument tabulatora ma wartość mniejszą od numeru aktualnej pozycji w wierszu - wystąpienie tabulatora nie powoduje żadnych efektów.

f/ Jeżeli ostatnim elementem instrukcji jest wyrażenie lub tabulator, instrukcja wyprowadzi jeszcze znaki końca wiersza, tak, że następna instrukcja drukowania będzie drukowała z nowej linii. Jeżeli instrukcję kończy przecinek lub średnik, znaki końca wiersza nie są wyprowadzane.

Jeżeli wydrukowanie kolejnego elementu powodowałoby przekroczenie marginesu, tj. liczbę znaków w wierszu większą od wartości marginesu danego kanału, wtedy system BASIC wyprowadzi sam znaki końca wiersza przed wyprowadzeniem danego elementu przy wyprowadzeniu wartości łańcuchowej w trakcie jej wyprowadzania.

Jak widać z powyższego opisu, drukowanie swobodne ma umożliwić jak najłatwiejsze programowanie wydruków wyjściowych, bez konieczności liczenia znaków, obawy o przekroczenie marginesu czy też o niewłaściwy dla danych liczb format.

Na następnych stronach zamieszczone są: wydruk z końcówki zawierający tekst programu oraz wyniki otrzymane przy wykonywaniu tego programu. Można na nich przeanalizować działanie drukowania swobodnego w BASIC-u.

Drukowanie według wzorca pozwala na zaspokojenie bardziej wyszukanych wymagań co do postaci wyników. Instrukcja drukowania, korzystająca z wzorca, zawiera słowo USING, po którym występuje numer wiersza lub wyrażenie łańcuchowe.

Wiersz programu, którego numer występuje po słowie USING w instrukcji drukowania musi mieć następującą budowę:

$\langle \text{wiersz wzorca} \rangle ::= \langle \text{numer wiersza} \rangle : \langle \text{łańcuch wzorca} \rangle$

WHO ARE YOU ? MARTIN.
CROOK-3 SB56
CG,UNPRI1. 159
*UNPRI1 CHR 5 5
BASIC.
*OLD UNPRI1

```
*LIST -1380
PROG UNPRI1
100 REM *** PROGRAM TO ILLUSTRATE UNFORMATTED PRINT ***
110 LET K=1
200 REM *** FUNCTION GENERATING INTEGERS ***
210 DEFFNI(M)=SGN(RND-.5)*(1+INT(2*RND*M-M))
220 REM *** FUNCTION GENERATING REALS ***
230 DEFFNR(N)=SGN(RND-.5)*(1+9*RND)*10^(SGN((RND-.5))*N*RND)
240 REM *** FUNCTION GENERATING INTEGERS AND REALS ***
250 REM --- P IS FRACTION OF INTEGERS BEING GENERATED ---
260 DEFFNL(P,M1,N1)
270 IF RND<P GOTO 300
280 FNL=FNR(N1)
290 GOTO 310
300 FNL=FNI(M1)
310 FNEND
320 REM *** FUNCTION GENERATING STRING OF LENGTH K ***
330 DEFFNS$(K)
340 FOR K=1 TO K
350 FNS$=FNS$&CHR$(65+26*RND)
360 NEXT K
370 FNEND
1180 REM --- PROGRAM PROPER BEGINS HERE ---
1190 PRINT "OUTPUT FILE ? ";
1200 INPUT F$
1205 FILE #K: F$
1210 IF PER(K,"PRINT")<1 STOP
1230 PRINT #K:
1240 PRINT "MARGIN SIZE ? ";
1250 INPUT M
1260 IF M>20 IF M<=80 GOTO 1285
1270 PRINT "WRONG MARGIN SIZE"
1280 GOTO 1240
1285 MARGIN #K: M
1290 LET N$="RESULTS OF PROGRAM "&JOB$
1300 PRINT #K: TAB(MRG(K)/2-LEN(N$)/2);N$
1310 LET D$="DATE : "&DAT$
1320 PRINT #K: USR$;TAB(MRG(K)-LEN(D$));D$
1325 PRINT #K: TAB(MRG(K)/2-6);"MARGIN :";MRG(K)
1330 GOSUB 2000
1340 FOR I=1 TO 15
1370 PRINT #K: FNL(.3,10000,20);
1380 NEXT I
*
```



```
*LIST 1381-
  ROG UNPRI1
1385 GOSUB 2000
1390 PRINT #K:
1395 GOSUB 2000
1400 FOR I=1 TO 8
1410 PRINT #K: FNL(.3,100000,35),
1412 NEXT I
1415 GOSUB 2000
1420 PRINT #K: LIN$(1)
1425 GOSUB 2000
1430 FOR I=1 TO 20
1440 PRINT #K: FNS$(5);"*";
1450 NEXT I
1460 GOSUB 2000
1465 PRINT #K:
1468 GOSUB 2000
1470 FOR I=1 TO 10
1480 PRINT #K: FNS$(10),
1490 NEXT I
1495 GOSUB 2000
1500 PRINT #K:
1510 PRINT #K: TAB(MRG(K)/2-5);"*** END ***"
1520 STOP
2000 REM *** ROUTINE TO MARK CHARACTER POSITIONS IN PRINT-OUT ***
2010 PRINT #K: ' TO O/P NEWLINE
2015 PRINT #K: MUL$(MRG(K),"~")
2020 FOR I=0 TO MRG(K)-1
2030 LET X=INT(I/10)
2040 IF X=I/10 GOTO 2070
2050 PRINT #K: " ";
2060 GOTO 2080
2070 PRINT #K: CHR$(48+MOD(X,10));
2080 NEXT I
2090 PRINT #K:
2100 FOR I=0 TO MRG(K)-1
2110 PRINT #K: CHR$(48+MOD(I,10));
2120 NEXT I
2130 PRINT #K:
2135 PRINT #K: MUL$(MRG(K),"~")
2140 RETURN ' *** END OF ROUTINE
20000 END
*
```


*RUN

RUNNING UNPRI1 1234
OUTPUT FILE ? MOO***

MARGIN SIZE ? 50

RESULTS OF PROGRAM UNPRI1

MARTIN

DATE : 07 MAY 81

MARGIN : 50

0 1 2 3 4
01234567890123456789012345678901234567890123456789

5743 5.14038E-13 3.06434E-17 -.247609
4.59730E+15 -3.18014E -3 -1.62033E+10 -1167
7.06004E-10 -6.30004E+11 -1.59679E+18 44.4193
-6.86447E +8 -1.01851E -9 3.27485E -4

0 1 2 3 4
01234567890123456789012345678901234567890123456789

0 1 2 3 4
01234567890123456789012345678901234567890123456789
-2.27451E-18 1.27018E -7 2.37358E-17
26443 -1.56519E-28 4.24852E-10
-4.58941E-27 -1.04215E+10

0 1 2 3 4
01234567890123456789012345678901234567890123456789

0 1 2 3 4
01234567890123456789012345678901234567890123456789
IQWFK*WHNJQ*LWYFV*ZUPZF*EQXEP*QROJQ*WGJKR*YDECE*IU
JCD*EMHJE*FRBQT*PFEGJ*XXEKM*UYEXO*NEDTL*UTDOQ*WOIG
K*JUPGJ*XXGWR*ABKNQ*

0 1 2 3 4
01234567890123456789012345678901234567890123456789

0 1 2 3 4
01234567890123456789012345678901234567890123456789
VDBLTIMWUV LAAWIJRXKK GQXOSCPIQF AJLKX
FIWJN JMOPSUSTRA POKQZJKDZJ RGAKT
GKZSC VBSUJCGACI JXEIMUNNCM DAGGF
LHKKN

0 1 2 3 4
01234567890123456789012345678901234567890123456789

*** END ***

STOP--AT 1520

*RUN

RUNNING UNPRI1 1234

OUTPUT FILE ? MOD***

MARGIN SIZE ? 60

RESULTS OF PROGRAM UNPRI1

MARTIN

DATE : 07 MAY 81

MARGIN : 60

0 1 2 3 4 5
01234567890123456789012345678901234567890123456789

5743 5.14038E-13 3.06434E-17 -.247609 4.59730E+15
-3.18014E -3 -1.62033E+10 -1167 7.06004E-10 -6.30004E+11
-1.59679E+18 44.4193 -6.36447E +8 -1.01851E -9
3.27485E -4

0 1 2 3 4 5
01234567890123456789012345678901234567890123456789

0 1 2 3 4 5
01234567890123456789012345678901234567890123456789

-2.27451E-18 1.27018E -7 2.37358E-17 26443
-1.56519E-28 4.24852E-10 -4.58941E-27 -1.04215E+10

0 1 2 3 4 5
01234567890123456789012345678901234567890123456789

0 1 2 3 4 5
01234567890123456789012345678901234567890123456789

IOWFK*WHNJQ*LWYFV*ZUPZF*EQXEP*GROJQ*WGJKR*YDECE*IUJCD*EMHJE*
FRBQT*PFEQI*XXEKM*UYEXO*NEDTL*UTDOQ*WOIGK*JUPGJ*XXGWR*ABKNQ*

0 1 2 3 4 5
01234567890123456789012345678901234567890123456789

0 1 2 3 4 5
01234567890123456789012345678901234567890123456789

VDBLTIMWUV LAAWIJRXKK GQXOSCPIQF AJLKXFIWJN
JMOPSUSTRA POKQZJKDZJ RGAKTGKZSC VBSUJCGACI
JXEIMUNNCM OAGQFLHKKK

0 1 2 3 4 5
01234567890123456789012345678901234567890123456789

*** END ***

STOP--AT 1520

*RUN
RUNNING UNPRI1 1234
OUTPUT FILE ? MOO***

MARGIN SIZE ? 72

RESULTS OF PROGRAM UNPRI1

MARTIN

DATE : 07 MAY 81

MARGIN : 72

0	1	2	3	4	5	6	7
01234567890123456789012345678901234567890123456789012345678901							

5743	5.14038E-13	3.06434E-17	-.247609	4.59730E+15	-3.18014E -3		
-1.62033E+10	-1167	7.06004E-10	-6.30004E+11	-1.59679E+18	44.4193		
-6.86447E +8	-1.01851E -9	3.27485E -4					

0	1	2	3	4	5	6	7
01234567890123456789012345678901234567890123456789012345678901							

0	1	2	3	4	5	6	7
01234567890123456789012345678901234567890123456789012345678901							

-2.27451E-18	1.27018E -7	2.37358E-17	26443				
-1.56519E-28	4.24852E-10	-4.58941E-27	-1.04215E+10				

0	1	2	3	4	5	6	7
01234567890123456789012345678901234567890123456789012345678901							

0	1	2	3	4	5	6	7
01234567890123456789012345678901234567890123456789012345678901							

IOWFK*WHNJQ*LWYFV*ZUPZF*EQXEP*QROJQ*WBJKR*YDECE*IUJCD*EMHJE*FRBQT*PFEDI*							
XXEKM*UYEXO*NEDTL*UTDOQ*WOTOK*JUPGJ*XXGWR*ABKNQ*							

0	1	2	3	4	5	6	7
01234567890123456789012345678901234567890123456789012345678901							

0	1	2	3	4	5	6	7
01234567890123456789012345678901234567890123456789012345678901							

VDBLTIMWUV	LAAWIJRXKK	GQXOSCPIQF	AJLKXFIWJN	JMOPUSUSTRA			
POKQZJKDZJ	RGAKTGKZSC	VBSUJCGACI	JXEIMUNNCH	OAGQFLHKKH			

0	1	2	3	4	5	6	7
01234567890123456789012345678901234567890123456789012345678901							

*** END ***

STOP--AT 1520

*

$$\langle \text{łańcuch wzorca} \rangle ::= [\langle \text{znak lit.} \rangle] \dots \left\{ \begin{array}{l} \langle \text{pole arytm.} \rangle \\ \langle \text{pole łańc.} \rangle \end{array} \right\} [\langle \text{znak lit.} \rangle] \dots$$

$$\left[\begin{array}{l} \langle \text{pole arytmetyczne} \rangle \\ \langle \text{pole łańcuchowe} \rangle \end{array} \right] \dots [\langle \text{znak literalny} \rangle] \dots$$

$$\langle \text{pole arytm.} \rangle ::= \left[\begin{array}{l} + \\ - \end{array} \right] \left\{ \begin{array}{l} \# [\#] \dots [\langle \text{kropka} \rangle \# [\#] \dots] \\ \langle \text{kropka} \rangle \# [\#] \dots \end{array} \right\} [\uparrow \uparrow \uparrow [\uparrow [\uparrow]]]$$

$$\langle \text{pole łańc.} \rangle ::= \left\{ \begin{array}{ll} \langle \text{znak "mniejszy"} \rangle & [\langle \text{znak "mniejszy"} \rangle] \dots \\ \langle \text{znak "większy"} \rangle & [\langle \text{znak "większy"} \rangle] \dots \end{array} \right\}$$

Jeżeli po słowie USING występuje wyrażenie łańcuchowe, to jego wartość jest traktowana jako łańcuch wzorca. Znaki literalne w łańcuchu wzorca mogą być dowolnymi znakami, za wyjątkiem znaków $\#$ $\langle \rangle$, zaś użycie znaków $+$ $-$ $.$ $\uparrow$ jest ograniczone - nie mogą być używane w sensie literalnym tam, gdzie mogłyby być wzięte za część pola.

Dwa pola mogą wystąpić obok siebie nie przedzielone znakami literalnymi w następujących przypadkach:

1. Pola są różnego typu - pierwsze arytmetyczne a drugie łańcuchowe lub odwrotnie
2. Jeżeli oba pola są arytmetyczne, to albo pierwsze musi się kończyć znakiem $\uparrow$ albo drugie zaczynać znakiem $+$ lub $-$
3. Jeżeli oba pola są łańcuchowe, to jedno musi się składać ze znaków $\langle$ a drugie ze znaków $\rangle$

Wykonanie instrukcji drukowanie według wzorca przebiega w następujący sposób: Wyprowadzane są znaki literalne poprzedzające pierwsze pole, dokładnie tak jak występują w łańcuchu wzorca. Następnie obliczana jest wartość pierwszego wyrażenia z listy, znaki ją reprezentujące zamieniają w odpowiedni sposób znaki pola i są wyprowadzane. Następnie

są wyprowadzane znaki literalne do początku następnego pola i w podobny sposób wartość drugiego wyrażenia z listy, itd. aż do wyczerpania listy wyrażeń. Następnie, jeśli lista wyrażeń nie kończy się średnikiem, wyprowadzane są znaki zmiany wiersza. Jeżeli łańcuch wzorca skończy się przed wyczerpaniem listy wyrażeń, będzie on interpretowany w podobny sposób od początku.

Wypełnianie znaków tworzących pole przebiega dla wartości arytmetycznej w następujący sposób: znaczące cyfry części całkowitej zastępują znaki numeru na lewo od kropki. Jeżeli liczba jest ujemna, najbardziej znaczącą cyfrę poprzedzi znak minus. Jeżeli pole zaczyna się znakiem + a liczba jest dodatnia, najbardziej znaczącą cyfrę poprzedzi znak +. Pozostałe miejsca pola na lewo od miejsca zajętego przez znak liczby lub przez najbardziej znaczącą cyfrę zostaną zastąpione spacjami. Znaki numeru znajdujące się w polu po kropce zostaną zastąpione częścią ułamkową liczby. Zera nieznaczące na końcu są również wyprowadzane. Ostatnia cyfra, wypełniająca ostatni znak numeru w polu jest wynikiem zaokrąglenia. Następnie, jeżeli w polu występują znaki ↑ są one zastępowane literą E oraz znakiem i wartością wykładnika. Jeżeli utworzona według powyższych zasad liczba nie pomieściłaby się w polu, system stosuje wydruk awaryjny, starając się przedstawić daną liczbę z mniejszą dokładnością i bez dbania już o właściwy format, ale na takiej samej liczbie pozycji znakowych z ilu składa się pole. Jeżeli pole jest za małe nawet dla wydruku awaryjnego, to całe pole jest zastępowane znakami gwiazdki *.

Jeżeli wyprowadzana jest wartość łańcuchowa, a pole składa się ze znaków < , to wydruk wartości jest wyrównywany w lewo, tzn. pierwszy znak wartości łańcuchowej zastąpi pierwszy znak < pola, drugi znak wartości drugi znak < pola itd. Jeśli długość wartości łańcuchowej jest mniejsza od długości pola, pozostałe znaki pola zostaną zastąpione spacjami. Jeżeli długość wartości łańcuchowej jest większa od długości pola, pozostałe znaki wartości

łańcuchowej zostaną zignorowane.

Wypełnienie pola złożonego ze znaków > przebiega podobnie, z tym, że wydruk wartości jest wyrównany w prawo, tzn. ostatni znak wartości łańcuchowej zastępuje ostatni znak > pola, przedostatni - przedostatni znak > itd. Jeżeli długość wartości łańcuchowej jest mniejsza od długości pola, pozostałe początkowe znaki pola są zastępowane spacjami. Jeżeli wartość łańcuchowa jest dłuższa od pola, pozostałe początkowe znaki wartości łańcuchowej są ignorowane.

Przy wykonywaniu instrukcji system, podobnie jak to już poprzednio omówiliśmy, kontroluje, czy nie grozi przekroczenie wartości marginesu, i, jeśli tak, to w odpowiednim miejscu wyprowadza sam znaki zmiany wiersza - tak jednak, by nie przedzieliły one wyprowadzanej liczby.

Na następnej stronie podano przykłady krótkich programów i wydrukowane przez nie wyniki. Pozwalają one zorientować się w sposobie działania drukowania według wzorca.

Instrukcja drukowania zmiennych indeksowanych służy do wyprowadzania całych zmiennych indeksowanych, według ich aktualnych wymiarów. Jej składnia jest następująca:

<instrukcja drukowania zmiennych indeksowanych> ::=

MAT PRINT[# <kanał>:] <zm.indeks.> [{';'} . <zm.indeks.>] ... [;]

W przypadku zmiennych indeksowanych arytmetycznych nie są wyprowadzane elementy o wskaźnikach równych zero, które są wyprowadzane dla zmiennych indeksowanych łańcuchowych. Macierze wyprowadzane są wierszami, po każdym wierszu macierzy wyprowadzane są znaki zmiany wiersza wydruku.

Odstępy między poszczególnymi elementami zależą od znaku za nazwą zmiennej indeksowanej - przecinek powoduje wydruk w strefach co 15 znaków, średnik - z minimalnymi odstępami, według zasad drukowania swobodnego. Brak przecinka lub średnika za ostatnią zmienną na liście działań przy macierzy tak, jakby tam był przecinek, zaś w przypadku wektora powoduje wydruk każdego elementu w oddzielnym wierszu.

Na następnej stronie podany jest program ilustrujący powyższe zasady i wyniki otrzymane przy wykonywaniu tego programu.


```
CG,MATPR2. 157
  *MATPR2 CHR 2 2
BASIC.
*OLD MATPR2
```

```
*LIST
PROG MATPR2
100 MARGIN 60
110 DIM A(12,5),B(20)
120 PRINT "WYMIARY : ";
130 INPUT M,N
140 LET M=ABS(M) ; N=ABS(N)
150 IF M>0 IF N>0 GOTO 290 ' IF MATRIX
160 MAT B=ZER(M+N)
170 MAT V$=NUL$(M+N)
180 FOR I=1 TO M+N
190 LET B(I)=I*10
200 NEXT I
210 PRINT MUL$(15,"****")
220 MAT PRINT B
230 PRINT MUL$(15,"****")
240 MAT PRINT B;
250 PRINT MUL$(15,"****")
260 MAT PRINT B;
270 PRINT MUL$(15,"****")
280 GOTO 120
290 REM ----- MATRIX -----
300 MAT A=ZER(M,N)
310 FOR I=1 TO M
320 FOR J=1 TO N
330 LET A(I,J)=1000*I+J
340 NEXT J
350 NEXT I
360 PRINT MUL$(15,"****")
370 MAT PRINT A
380 PRINT MUL$(15,"****")
390 MAT PRINT A;
400 PRINT MUL$(60,"*")
410 GOTO 120
420 END
*RUN
```

```
RUNNING MATPR2 851
WYMIARY : 0,2
```

```
*****
```

```
10
20
```

```
*****
```

```
10          20
```

```
*****
```

```
10 20
```

```
*****
```

```
WYMIARY :
```


23,5

1001	1002	1003	1004
1005			
2001	2002	2003	2004
2005			
3001	3002	3003	3004
3005			

1001	1002	1003	1004	1005
2001	2002	2003	2004	2005
3001	3002	3003	3004	3005

WYMIARY : 5,3

1001	1002	1003
2001	2002	2003
3001	3002	3003
4001	4002	4003
5001	5002	5003

1001	1002	1003
2001	2002	2003
3001	3002	3003
4001	4002	4003
5001	5002	5003

WYMIARY : 3,8

1001	1002	1003	1004
1005	1006	1007	1008
2001	2002	2003	2004
2005	2006	2007	2008
3001	3002	3003	3004
3005	3006	3007	3008

1001	1002	1003	1004	1005	1006	1007	1008
2001	2002	2003	2004	2005	2006	2007	2008
3001	3002	3003	3004	3005	3006	3007	3008

WYMIARY : 0,3

10
20
30

10 20 30

10 20 30

WYMIARY :

27.15

MACIERZ ZBYT DUZA--AT 300
300 MAT A=ZER(M,N)
*RUN
RUNNING MATPR2 851
WYMIARY : 5,15

MACIERZ ZBYT DUZA--AT 300
300 MAT A=ZER(M,N)
*RUN
RUNNING MATPR2 851
WYMIARY : 4,15

MACIERZ ZBYT DUZA--AT 300
300 MAT A=ZER(M,N)
*4,14
NIEROZPOZNANA INSTRUKCJA--AT 4
*RUN
RUNNING MATPR2 851
WYMIARY : 4,14

1001	1002	1003	1004
1005	1006	1007	1008
1009	1010	1011	1012
1013	1014		
2001	2002	2003	2004
2005	2006	2007	2008
2009	2010	2011	2012
2013	2014		
3001	3002	3003	3004
3005	3006	3007	3008
3009	3010	3011	3012
3013	3014		
4001	4002	4003	4004
4005	4006	4007	4008
4009	4010	4011	4012
4013	4014		

1001	1002	1003	1004	1005	1006	1007	1008	1009	1010
1011	1012	1013	1014						
2001	2002	2003	2004	2005	2006	2007	2008	2009	2010
2011	2012	2013	2014						
3001	3002	3003	3004	3005	3006	3007	3008	3009	3010
3011	3012	3013	3014						
4001	4002	4003	4004	4005	4006	4007	4008	4009	4010
4011	4012	4013	4014						

WYMIARY : 0,50

MACIERZ ZBYT DUZA--AT 160
160 MAT B=ZER(M+N)
*1RUN
RUNNING MATPR2 851
WYMIARY : 1,3

1001	1002	1003
------	------	------

1001	1002	1003
------	------	------

WYMIARY :

OPERACJE NA ZBIORACH O DOSTĘPIE SWOBODNYM

Zbiory o dostępie swobodnym (zbiory binarne - ABS) zbudowane są inaczej niż zbiory znakowe. Można sobie wyobrazić, że zawierają one ponumerowane elementy, z których każdy przechowuje jedną wartość. W zależności od typu wartości możemy podzielić je na zbiory arytmetyczne i zbiory łańcuchowe.

Niepusty zbiór o dostępie swobodnym o długości n zawiera n elementów, ponumerowanych od zera do $n-1$, n -ty element zawiera znacznik końca zbioru.

Dostęp do zbioru w czasie wykonywania programu uzyskuje się za pomocą opisanej już poprzednio instrukcji otwarcia kanału. Po wykonaniu tej instrukcji wszelka komunikacja z danym zbiorem odbywa się poprzez numer kanału.

Z każdym otwartym niepustym zbiorem o dostępie swobodnym są związane następujące wielkości:

- Długość zbioru - liczba elementów zbioru. Jej wartości można w każdej chwili użyć w programie poprzezwołanie funkcji standardowej $LOF(K)$, gdzie K jest wyrażeniem wskaźnikowym określającym numer kanału przypisany do danego zbioru.
- Wskaźnik zbioru - jest on numerem elementu, którego będzie dotyczyła najbliższa operacja we/wy. Wartość tego wskaźnika ma funkcja $LOC(K)$.
- Margines. Dla zbioru arytmetycznego wynosi 1. Dla zbioru łańcuchowego oznacza długość elementu zbioru w znakach. Poszczególne elementy mogą być krótsze od wartości marginesu. Funkcja $MKG(K)$ ma wartość marginesu zbioru na kanale K .

Instrukcja ustawiania marginesu, którą poznaliśmy już uprzednio, może być wykonana wyłącznie na zbiorze nowo otwartym, zbiorze pustym bądź zbiorze, którego zawartość wymazano za pomocą instrukcji wymazania zawartości zbioru o składni:

<instrukcja wymazania zawartości zbioru> ::=

SCRATCH # <kanał>:

Instrukcji tej można również używać dla zbiorów znakowych

Instrukcja ustawiania wskaźnika zbioru służy do zmiany wartości wskaźnika zbioru o dostępie swobodnym i ma następującą postać:

<instrukcja ustawiania wskaźnika zbioru> ::=

RESET # <kanał>: <wyrażenie wskaźnikowe>

Wiersz programu:

200 RESET #K :

oraz wiersz:

200 RESET #K : 0

są równoważne z punktu widzenia działania programu.

Próbie ustawienia wskaźnika na wartość większą od długości zbioru system wykrywa jako błąd.

Bezpośrednio po wykonaniu instrukcji otwarcia kanału FILE dla zbioru o dostępie swobodnym, wskaźnik zbioru jest inicjalizowany w taki sposób, że pierwsza instrukcja czytania lub zapisu ustawi go na zero, chyba, że przed operacjami czytania lub zapisu zostanie wykonana instrukcja ustawiania wskaźnika zbioru.

Do wprowadzania wartości ze zbiorów o dostępie swobodnym służy instrukcja czytania o następującej składni:

<instrukcja czytania> ::=

READ # <kanał>: { <zmienna arytmetyczna> [, <zmienna arytmetyczna>] ... }
{ <zmienna łańcuchowa> [, <zmienna łańcuchowa>] ... }

Przy wykonywaniu tej instrukcji pobierany jest element zbioru binarnego o numerze określonym aktualną wartością wskaźnika zbioru na danym kanale i wartość tego elementu

nadawana jest pierwszej zmiennej z listy, poczem wartość wskaźnika zbioru zwiększana jest o 1. Następnie te same czynności powtarzane są dla następnych zmiennych z listy, aż do jej wyczerpania.

Jeżeli w czasie wykonywania programu nastąpi próba wczytania znacznika końca zbioru, wykonanie programu zostanie zakończone z odpowiednim wydrukiem. Typ zmiennych na liście musi być zgodny z typem zbioru.

Instrukcja zapisu umożliwia zapisanie wartości do elementów zbioru o dostępie swobodnym, z usunięciem poprzedniej zawartości, bądź dopisywanie elementów do zbioru o dostępie swobodnym (rozszerzanie zbioru). Jej składnia jest następująca:

<instrukcja zapisu> ::=

$$\text{WRITE \# \langle kanał \rangle : } \left\{ \begin{array}{l} \langle \text{wyr. arytm.} \rangle \left[, \langle \text{wyr. arytm.} \rangle \right] \dots \\ \langle \text{wyr. łańcuch.} \rangle \left[. \langle \text{wyr. łańcuch.} \rangle \right] \dots \end{array} \right\}$$

Przy wykonywaniu tej instrukcji wskaźnik zbioru ma wartość mniejszą od lub równą długości zbioru. Wartości wyrażeń z listy są kolejno obliczane i zapisywane do zbioru zgodnie z wartością wskaźnika, który po zapisaniu każdej wartości jest zwiększany o jedność. Jeżeli przed zapisaniem kolejnej wartości wskaźnik zbioru będzie równy długości zbioru, nastąpi rozszerzenie zbioru o jeden element.

Typ wyrażeń na liście musi być zgodny z typem zbioru.

W podobny sposób do instrukcji czytania i zapisu działają instrukcje czytania i zapisu zmiennych indeksowanych pozwalające czytać i zapisywać całe zmienne indeksowane.

<instrukcja czytania zmiennych indeksowanych> ::=

MAT READ # <kanał> :

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} \langle \text{mac. arytm.} \rangle [(\langle \text{w.w.} \rangle , \langle \text{w.w.} \rangle)] \\ \langle \text{wekt. arytm.} \rangle [(\langle \text{w.w.} \rangle)] \end{array} \right\} \left[\left\{ \begin{array}{l} \langle \text{mac. arytm.} \rangle [(\langle \text{w.w.} \rangle , \langle \text{w.w.} \rangle)] \\ \langle \text{wekt. arytm.} \rangle [(\langle \text{w.w.} \rangle)] \end{array} \right\} \dots \right] \\ \left\{ \begin{array}{l} \langle \text{mac. łańc.} \rangle [(\langle \text{w.w.} \rangle , \langle \text{w.w.} \rangle)] \\ \langle \text{wekt. łańc.} \rangle [(\langle \text{w.w.} \rangle)] \end{array} \right\} \left[\left\{ \begin{array}{l} \langle \text{mac. łańc.} \rangle [(\langle \text{w.w.} \rangle , \langle \text{w.w.} \rangle)] \\ \langle \text{wekt. łańc.} \rangle [(\langle \text{w.w.} \rangle)] \end{array} \right\} \dots \right] \end{array} \right\}$$

<instrukcja zapisu zmiennych indeksowanych> ::=

MAT WRITE # <kanal>:

$$\left\{ \begin{array}{ll} \langle \text{zmienna indeks.arytm.} \rangle [, \langle \text{zmienna indeks.arytm.} \rangle] & \dots \\ \langle \text{zmienna indeks.łańc.} \rangle [, \langle \text{zmienna indeks.łańc.} \rangle] & \dots \end{array} \right\}$$

Poniżej podano wydruk programu, stosującego instrukcje czytania i zapisu. Program ten działa na zbiorze o 10000 elementów. Jeżeli zbiór nie zawiera tylu elementów, lub nie jest zbiorem arytmetycznym, program go inicjalizuje zapisuje doń 10000 elementów zerowych, używając instrukcji MAT WRITE (bo to szybciej niż po 1 elemencie). Następnie uzupełnia znajdujący się w tym zbiorze rozkład częstości o nowych 10000 elementów, generowanych za pomocą funkcji RND.

Czas wykonywania takiego programu nie jest, niestety, krótki ze względu na znaczną liczbę operacji dyskowych.

Chcąc uzyskać szybsze wykonywanie programów należy tak programować operacje we/wy, by kolejne operacje dotyczyły elementów zbioru znajdujących się blisko siebie.

SUBPROGRAMY

Subprogramy stanowią oddzielne jednostki programowe, w których wszystkie obiekty są lokalne (numery wierszy, zmienne, bloki danych, kanały), a komunikacja z programem lub subprogramem wołającym odbywa się jedynie poprzez parametry. Funkcje systemowe, np. DET, ROW, COL, LOL itp. mają charakter nielokalny.

Przy pracy w trybie programowym można przejść do kontekstu subprogramu zleceniem NAME z podaniem nazwy subprogramu. Znakiem nasłuchu systemu w tym kontekście jest znak numeru #.

Subprogram musi się rozpoczynać wierszem nagłówka subprogramu i kończyć wierszem końca subprogramu. Między tymi wierszami mogą się znajdować dowolne wiersze programu

BASIC.
*OLD DISTRI

```
*LIST
PR06 DISTRI
100 DIM X(100) ' 31/3/81
110 PRINT "NO OF INTERVALS ? ";
120 INPUT K
130 LET K=INT(ABS(K))
140 IF K<10 GOTO 110
150 PRINT "NO. OF SAMPLES ? ";
160 INPUT S
170 LET S=INT(ABS(S))
180 FILE #1: "DISTR"
190 IF PER(1,"WRITE")=1 IF LOF(1)=K GOTO 280
200 RESET #1:
210 FOR I=1 TO INT(K/100)
220 MAT WRITE #1: X
230 NEXT I
240 FOR I=100*I TO K-1
250 WRITE #1: 0
260 NEXT I
270 PRINT LOF(1)
280 LET T=TIM'
290 FOR I=0 TO S-1
300 RESET #1: INT(RND*K)
310 READ #1: N
320 RESET #1: LOC(1)-1
330 WRITE #1: N+1
340 IF FRA((I+1)/100)<>0 GOTO 360
350 PRINT TIM-T,I+1
360 NEXT I
370 PRINT TIM-T,I+1
380 STOP
390 END
*BYE
```

BASIC FINISH 02570
CF,DISTR,ABS,-1.
CG,. 157
DISTR ABS 256.00

```
*RUN
RUNNING DISTRI 800
NO OF INTERVALS ? 10000
NO. OF SAMPLES ? 300
10000
32 100
64 200
95 300
95 300
```

STOP--AT 380
*BYE

BASIC FINISH 02570
CG,. 157
DISTR ABS 256.072462
TF,.
CG,. 157
DISTR ABS 118.072462
ES.
WHO ARE YOU ?

dotychczas opisane, za wyjątkiem

- wiersza końca programu
- deklaracji wymiarów dotyczących parametrów formalnych
- definicji funkcji będących parametrami formalnymi.

Wiersz końca subprogramu ma następującą składnię:

$\langle \text{wiersz końca subprogramu} \rangle ::=$

$\langle \text{nr wiersza} \rangle \text{ SUBEND } [' \langle \text{komentarz} \rangle]$

zaś wiersz nagłówka subprogramu taką jak poniżej:

$\langle \text{wiersz nagłówka subprogramu} \rangle ::=$

$\langle \text{nr wiersza} \rangle \langle \text{nagłówek subprogramu} \rangle [' \langle \text{komentarz} \rangle]$

$\langle \text{nagłówek subprogramu} \rangle ::=$

$\text{SUB } \langle \text{identyfikator} \rangle (\langle \text{par. form. subprogr.} \rangle [, \langle \text{par. for. sub.} \rangle] \dots)$

$\langle \text{parametr formalny subprogramu} \rangle ::=$

$$\left\{ \begin{array}{l} \langle \text{zmienna prosta} \rangle \\ \langle \text{macierz formalna} \rangle \\ \langle \text{wektor formalny} \rangle \\ \langle \text{funkcja definiowana} \rangle \end{array} \right\}$$

Nazwa subprogramu (identyfikator) musi być taka sama jak nazwa podzbioru roboczego ustalona zleceniem NAME.

Wykonanie subprogramu następuje wyłącznie w wyniku wykonania w programie (głównym lub w innym subprogramie)

instrukcji wołania subprogramu, której składnia opisana jest poniżej

$\langle \text{instrukcja wołania subprogramu} \rangle ::=$

$\text{CALL } \langle \text{identyf.} \rangle (\langle \text{par. akt. subprogr.} \rangle [, \langle \text{par. aktualny sub.} \rangle] \dots)$

$\langle \text{parametr aktualny subprogramu} \rangle ::=$

$$\left\{ \begin{array}{l} \langle \text{parametr aktualny subprogramu wołany przez wartość} \rangle \\ \langle \text{parametr aktualny subprogramu wołany przez nazwę} \rangle \end{array} \right\}$$

$\langle \text{parametr aktualny subprogramu wołany przez wartość} \rangle ::=$
 $\langle \langle \text{wyrażenie arytmetyczne} \rangle \rangle$

$\langle \text{parametr aktualny subprogramu wołany przez nazwę} \rangle ::=$
 $\left\{ \begin{array}{l} \langle \text{zmienna prosta} \rangle \\ \langle \text{macierz formalna} \rangle \\ \langle \text{wektor formalny} \rangle \\ \langle \text{funkcja definiowana} \rangle \end{array} \right\}$

Identyfikator jest nazwą subprogramu, pozwalającą wybrać odpowiedni subprogram przy wykonywaniu instrukcji wołania subprogramu. Liczba, typ i kolejność parametrów aktualnych muszą być zgodne z parametrami formalnymi w nagłówku subprogramu.

Wykonywanie instrukcji wołania subprogramu rozpoczyna się od inicjalizacji subprogramu:

- zerowane są wszystkie zmienne subprogramu, za wyjątkiem zmiennych będących parametrami formalnymi, których odpowiedniki w instrukcji wołania subprogramu są wołane przez nazwę,
- inicjalizowany jest blok danych subprogramu - wskaźnik danych ustawiany jest na pierwszą daną bloku,
- obliczane są wartości wyrażeń dla parametrów wołanych przez wartość i ich wartości nadawane zmiennym subprogramu będącym odpowiednimi parametrami formalnymi.

Następnie wykonywany jest subprogram, przy czym:

- nadanie wartości zmiennej prostej lub indeksowanej będącej parametrem formalnym, której odpowiadający parametr aktualny był wołany przez nazwę, spowoduje nadanie wartości zmiennej w programie wołającym będącej parametrem aktualnym. Jeżeli jest to zmienna arytmetyczna prosta, nadanie wartości zmiennej w programie wołającym nastąpi dopiero po zakończeniu wykonywania subprogramu (oczywiście, wartości wynikającej z ostatnio w subprogramie wykonanego przypisania wartości tej zmiennej);

Mechanizm ten umożliwia przesyłanie wyników do programu lub subprogramu wołającego.

- wykonanie instrukcji otwarcia zbioru, w której wyrażenie łańcuchowe jest nazwą zbioru otwartego w programie wołającym, spowoduje uzyskanie dostępu poprzez numer kanału lokalny w subprogramie do kanału w programie wołającym. Instrukcja taka nie wykonuje żadnych operacji na kanale czy zbiorze, a tylko umożliwia dostęp do kanału w programie wołającym. Podobnie, po wyjściu z subprogramu kanał będzie się znajdował w takim stanie dla programu wołającego, w jakim go pozostawił subprogram. Mechanizm ten pozwala w pośredni sposób przesyłać jako parametry kanały, poprzez użycie jako parametru aktualnego zmiennej łańcuchowej zawierającej nazwę zbioru.
- wykonanie instrukcji otwarcia kanału dla zbioru nie otwartego spowoduje otwarcie kanału (i zbioru). Przy zakończeniu wykonywania subprogramu i powrocie do programu wołającego kanały takie są zamykane.
- wystąpieniewołania funkcji definiowanej będącej parametrem spowoduje przesłanie argumentów funkcji z subprogramu i użycie do obliczania wartości funkcji odpowiedniej do parametru definicji z programu wołającego. Jeżeli definicja używa do obliczenia wartości funkcji zmiennych nielokalnych dla bloku definicji - zostają użyte ich aktualne wartości z programu wołającego.
- instrukcja czytania danych i instrukcje ustawiania wskaźnika danych dotyczą lokalnego bloku danych subprogramu.
- przewymiarowanie w subprogramie zmiennej indeksowanej będącej parametrem formalnym spowoduje przewymiarowanie zmiennej indeksowanej w programie wołającym, będącej odpowiednim parametrem aktualnym.

Napotkanie w czasie wykonywania subprogramu wiersza końca subprogramu spowoduje zakończenie wykonywania subprogramu i powrót do programu wołającego, do wiersza następującego


```

*OLD EXMAP
NEW=MAP

*CAT
PROG EXMAP
MAP :100 SUB MAP(X1,X9,Y1,Y9,Z1,Z9,FNZ,F$)
*LIST
PROG EXMAP
100 DEFFNF(X,Y)=(X-.5)*(X-.5)+Y*Y
110 PRINT "OUTPUT FILE ? ";
120 INPUT X$
130 FILE #1: X$
140 PRINT #1: "RESULTS OF ";JOB$;" OBTAINED AT ";DAT$
150 PRINT #1:
160 CALL MAP((( -1)),(1),(( -1)),(1),(0),(1),FNF,X$)
170 END
*NAME MAP
#LIST
NAME MAP
100 SUB MAP(X1,X9,Y1,Y9,Z1,Z9,FNZ,F$)
110 DATA 40 ' X SCALE ( NO. OF DISCRETE POINTS)
120 DATA 40 ' Y SCALE
130 DATA 10 ' Z DISCRETIZATION
140 READ P1,Q1,R1
150 FILE #2: F$
160 LET X5=(X9-X1)/P1 ; Y5=(Y9-Y1)/Q1 ; Z5=(Z9-Z1)/R1
170 FOR J=Q1 TO 0 STEP (-1)
180 FOR I=0 TO P1-1
190 LET Z=INT(FNZ(X1+I*X5,Y1+J*Y5)/Z5)
200 PRINT #2: USING "# ": MOD(Z,10);
210 NEXT I
220 PRINT #2:
230 NEXT J
240 SUB END
#LIST100
NAME MAP
100 SUB MAP(X1,X9,Y1,Y9,Z1,Z9,FNZ,F$)
#NAME EXMAP
*LIST100
PROG EXMAP
100 DEFFNF(X,Y)=(X-.5)*(X-.5)+Y*Y
*
```



```
*RUN
  RUNNING EXMAP 632
  OUTPUT FILE ? MOO***
  RESULTS OF EXMAP OBTAINED AT 06 MAY 81
```

[illegible]

END--AT 170

✱

*BYE

BASIC FINISH 02570

po wierszu zawierającym instrukcję wołania subprogramu. Takie samo działanie ma instrukcja, składająca się ze słowa SUBEXIT.

Jako przykład podany jest subprogram drukujący mapę wartości funkcji dwóch zmiennych w postaci cyfr oznaczających zawieranie się wartości funkcji w jednym z podprzedziałów wynikających z podziału podanego parametrami zakresu wartości na dziesięć części, ponumerowanych od zera do 9. Program główny powoduje wydruk na drukarce takiej mapy dla funkcji $z = (x - 0,5)^2 + y^2$ dla x z przedziału $-1,1$ i y z przedziału $-1,1$.

Cyfra 9 odpowiada wartości funkcji 0.9, 1.9; cyfra 0 wartościom funkcji 0.0, 1.0, 2.0.

INSTRUKCJA RANDOMIZACJI

Funkcja standardowa RND ma wartość, w kolejnych wołaniach, liczb pseudolosowych o rozkładzie równomiernym. W kolejnych wykonaniach programu generuje się zawsze taki sam ciąg liczb, co jest wygodne przy testowaniu programów. Zastosowanie instrukcji randomizacji, składającej się ze słowa RANDOMIZE, pozwala przy każdym wykonaniu programu uzyskać inny ciąg liczb pseudolosowych, dzięki temu, że pierwsza liczba tworzona jest przy użyciu informacji z zegara maszyny. Poniższy program wygeneruje za każdym wykonaniem inny ciąg liczb całkowitych o wartościach od 1 do 6 - może symulować ciągi rzutów kostką sześcienną:

BASIC.

*OLD DICE

*LIST

PROG DICE

100 MARGIN 50

110 RANDOMIZE

120 PRINT "HOW MANY ? ";

130 INPUT N

140 FOR I=1 TO N

150 PRINT INT(1+6*RND);

160 NEXT I

170 END

*RUN

RUNNING DICE 263

HOW MANY ? 20

3 4 4 2 2 5 5 2 5 6 4 5 4 2 2 3
3 3 5 3

END--AT 170

*RUN

RUNNING DICE 263

HOW MANY ? 20

5 3 5 3 5 1 3 6 6 3 6 4 6 2 4 5
2 1 3 6

END--AT 170

*RUN

RUNNING DICE 263

HOW MANY ? 50

2 3 6 4 2 2 1 5 1 6 2 2 2 2 6 6
1 4 6 4 2 6 3 6 1 3 4 3 1 2 6 5
3 1 4 4 1 3 3 4 5 5 2 6 6 6 2 3
2 3

END--AT 170

*RUN

RUNNING DICE 263

HOW MANY ? 32

1 4 6 2 5 3 6 5 6 4 5 1 2 5 6 1
2 1 1 3 5 4 3 2 6 3 2 1 5 3 3 6

END--AT 170

*RUN

RUNNING DICE 263

HOW MANY ? 32

4 4 1 2 2 5 4 3 1 1 1 6 4 5 2 2
1 4 5 6 5 4 6 4 4 5 3 3 1 5 3 3

END--AT 170

*RUN

RUNNING DICE 263

HOW MANY ? 32

6 3 2 3 5 6 2 1 2 4 3 5 1 5 4 4
5 1 2 3 6 5 3 6 2 1 4 4 3 2 4 5

END--AT 170

*RUN

RUNNING DICE 263

HOW MANY ? 32

2 3 6 5 6 4 2 2 5 1 2 1 1 5 3 1
2 5 6 1 2 5 5 1 2 1 6 3 2 4 6 5

END--AT 170

*RUN

RUNNING DICE 263

HOW MANY ?

STOP--AT 130

*BYE

BASIC FINISH 02570

INSTRUKCJA ZMIANY PROGRAMU (NAKLADKOWANIA)

Dość częste są sytuacje, że program, stopniowo rozbudowywany i uzupełniany, w pewnym momencie przestaje się mieścić w pamięci. Czasem pomocą dokładne przeanalizowanie i zmiana konstrukcji programu, usunięcie komentarzy, zmniejszenie rozmiarów używanych zmiennych indeksowanych. Jeżeli to nie wystarcza, trzeba program podzielić na dwa lub więcej oddzielnych programów. Często z góry wiadomo, że program będzie bardzo duży i wtedy warto go od razu rozbić na dwa lub kilka mniejszych, kierując się przy podziale logiką i funkcjonalnością.

Instrukcja zmiany programu służy do przekazywania sterowania innemu programowi i daje odpowiednie mechanizmy przekazywania informacji między tymi programami. Jej składnia jest taka jak poniżej:

CHAIN <wyr.łańc.> : [<zm.indeks.> [, <zm.indeks.>] ...]

Wyrażenie łańcuchowe po słowie CHAIN musi mieć taką samą budowę jak w instrukcji otwarcia kanału, tj.

$$\left\{ \begin{array}{l} \langle \text{nazwa zbioru} \rangle \\ \langle \text{urządzenie wspólne} \rangle \end{array} \right\}$$

Działanie instrukcji polega na tym, że ze zbioru bądź urządzenia pobierany jest nowy program, który ma zastąpić program aktualnie wykonywany (stary program). Program ten może być albo w postaci źródłowej, albo (tylko już w zbiorze, typu RBS) w postaci obrazu pamięci. Stary program, wraz z subprogramami, zostaje usunięty z pamięci. Nowy program zostaje umieszczony w pamięci, w miarę potrzeby jest sprawdzana jego poprawność, poczem jest inicjalizowany i uruchamiany. W trakcie inicjalizacji następuje przekazanie informacji ze starego programu. Inicjalizacja polega między innymi na wykonaniu poniższych czynności:

- wszystkim zmiennym prostym, które występują w nowym programie, a nie występowały w starym, nadawana jest wartość zero lub łańcucha pustego. Zmienne proste, które występują w obu programach będą miały po rozpoczęciu wykonywania nowego programu takie same wartości, jakie miały ostatnio w starym programie.
- zmienne indeksowane, występujące na liście po znaku dwukropka są przenoszone do nowego programu z aktualnymi wymiarami, z tym, że wymiary aktualne stają się rozmiarami - tak, jakby w nowym programie znajdowała się deklaracja wymiarów według aktualnych wymiarów. Wartości elementów tych zmiennych w nowym programie będą takie same jak ostatnio w starym programie. Pozostałe zmienne indeksowane nowego programu są inicjalizowane w zwykły sposób - ich wszystkim elementom nadawane są wartości zero lub łańcucha pustego.
- do nowego programu przenoszona jest informacja o otwartych w starym programie kanałach - kanały będą dostępne w nowym programie poprzez te same numery co w starych a ich stan nie ulegnie żadnej zmianie w wyniku działania instrukcji zmiany programu.

Poza powyższymi czynnościami inicjalizacja odbywa się tak jak przy uruchamianiu programu zleceniem RUN. Po inicjalizacji nowy program zostaje uruchomiony od wiersza o najniższym numerze.

INSTRUKCJA DOŁĄCZANIA Z BIBLIOTEKI

Jeżeli w programie występują wołania subprogramów, to w czasie wykonywania programu wołane subprogramy muszą znajdować się w pamięci wraz z programem wołającym. Do automatycznego dołączania subprogramów w czasie uruchamiania programu służy omawiana właśnie instrukcja o następującej składni:

SET TIME 9,33,40.

SET TIME 9,34,30. 7 MAY 1981

WHO ARE YOU ? TESTER,.

CROOK-3 SB56

CG,CALC. 29

*CALC CHR 1 1

BASIC.

*OLD CALC

*LIST

PROG CALC

10 FILE #1: JOB\$

20 FILE #2: "CALWRK"

30 RESET #1:

40 RESET #2:

50 PRINT "PRINT : ";

60 INPUT X\$

70 PRINT #2: "10 PRINT "&X\$

80 INPUT #1: X\$,X\$,X\$

90 IF PER(1,"INPUT")=0 GOTO 130

100 INPUT #1: X\$

110 PRINT #2: X\$

120 GOTO 90

130 RESET #2:

140 CHAIN "CALWRK":

150 END

*BYE

BASIC FINISH 02570

CG,CALWRK. 29

CG,CALWRK. 29

*RUN

RUNNING CALC 318

PRINT : CLK\$

RUNNING CALC 557

09:36:24

PRINT : CUF\$

RUNNING CALC 557

CALC

PRINT : DAT\$

RUNNING CALC 557

07 MAY 81

PRINT : JOB\$

RUNNING CALC 557

CALC

PRINT : SYS\$

RUNNING CALC 557

BASIC-EXT MERA 400 ISS.1 03.02.80

PRINT :

BASIC STOPPED 07576


```
RS. RUNNING CALC 318
PRINT : @PI
  RUNNING CALC 557
  3.14159
PRINT : USING "###.#####":@PI
  RUNNING CALC 566
  3.1415926536
PRINT : RND
  RUNNING CALC 557
  .189282
PRINT : RND
  RUNNING CALC 557
  .189282
PRINT : EPS
  RUNNING CALC 557
  1.81899E-12
PRINT : INF
  RUNNING CALC 557
  1.70141E+38
PRINT : INX
  RUNNING CALC 557
  32767
PRINT : NIL
  RUNNING CALC 557
  1.46937E-39
PRINT : VAL(AAA123
POMIESZANIE OPERACJI LANCUCHOWYCH I ARYTMETYCZNYCH--AT 10
10 PRINT VAL(AAA123
*BYE
```

```
BASIC FINISH 02570
CG,CALWRK\ 29
  CALWRK CHR 32 1
LISM.? CALWRK
```

```
10 PRINT VAL(AAA123
30 RESET #1:
40 RESET #2:
50 PRINT "PRINT : ";
60 INPUT X$
70 PRINT #2: "10 PRINT "&X$
80 INPUT #1: X$,X$,X$
90 IF PER(1,"INPUT")=0 GOTO 130
100 INPUT #1: X$
110 PRINT #2: X$
120 GOTO 90
130 RESET #2:
140 CHAIN "CALWRK":
150 END
```

LISM DELETED

BASIC.
*OLD CALC

*


```
*RUN
  RUNNING CALC 318
PRINT : VAL("AAA123BBBCC"),PVL
  RUNNING CALC 568
123      7
PRINT : VAL("123 ABC"),PVL
  RUNNING CALC 566
123      5
PRINT : VAL("123 ABC"),PVL
  RUNNING CALC 566
123      5
PRINT : VAL("123ABC"),PVL
  RUNNING CALC 565
123      4
PRINT : VAL("123      "),PVL
  RUNNING CALC 569
123      5
PRINT : COD(1,"ABC")
  RUNNING CALC 562
65
PRINT : COD(2,"ABC")
  RUNNING CALC 562
66
PRINT : DAY
  RUNNING CALC 557
7
PRINT : DYR
  RUNNING CALC 557
127
PRINT : MON
  RUNNING CALC 557
5
PRINT : MRG(0)
  RUNNING CALC 558
150
PRINT : PER(0,"INPUT")
  RUNNING CALC 563
?
1
  RUNNING CALC 555
PRINT :
PRINT : PER(1,"FILE")
  RUNNING CALC 562
0
PRINT : PER(2,"FILE")
  RUNNING CALC 562
0
PRINT : PER(3,"FILE")
  RUNNING CALC 566
1
PRINT : PER(3,"XXX")
  RUNNING CALC 566
-1
PRINT :
```



```

?
?PER(1,("READ"))
RUNNING CALC 562
0
PRINT : MUL$(10,"*")
RUNNING CALC 565
*****
PRINT : MUL$(10,"--")
RUNNING CALC 566
-----
PRINT : USING MUL$(10,"#"):I/EP$
RUNNING CALC 574
0
PRINT : USING MUL$(10,"#"):1/EP$
RUNNING CALC 570
549756E +6
PRINT : USING MUL$(11,"#"):1/EP$
RUNNING CALC 570
5497558E +5
PRINT : USING MUL$(12,"#"):1/EP$
RUNNING CALC 570
549755813887
PRINT : INT(1.9999999)
RUNNING CALC 562
1
PRINT : INT(-1.00000001)
RUNNING CALC 563
-2
PRINT :
STOP--AT 60
*LIST60
PROG CALC
60 LINPUT X$
*NLI60
60 LINPUT X$
*N L I 6 0
ZLECENIE NIEDOPUSZCZALNE DLA PROGRAMU WYNIKOWEGO
*NLI 6 0
ZLECENIE NIEDOPUSZCZALNE DLA PROGRAMU WYNIKOWEGO
*NLI 60
60 LINPUT X$
* N L I 60
60 LINPUT X$
* N L I 6 0
60 LINPUT X$
*BYE

ASIC FINISH 02570
CG,CALWRK. 29
CALWRK CHR 32 1
PF,CALWRK. 28
CG,. 28
*CALWRK CHR 1 1
NP,.
CG,. 28
*CALWRK CHR 1 1
EF,. 29
CG,CALWRK. 29

```



```
BASIC.
*NEW INTM
*LIST
*
BASIC STOPPED 01435
.
*50 FILE#1:JOB$
*60 FILE #2:"TIWRK"
*70 RESET#1:
*80 RESET#2:
*90 IF L0/=<>0 GOTO 190
*NLI90
  90 IF L0<>0 GOTO 190
*100 L0=L1=1000
*110 T0=TIM
*120 FOR L0=1 TO L0
*130 NEXT L0
*140 T0=TIM-T0
*150 IF T0=0 LET L0=L1=L0*10 GOTO 110
*160 IF T0>=100 IF T0<200 GOTO 190
*170 L0=L1=INT(L0*100/T0)+1
*180 GOTO 110
*190 IF T1=0 GOTO 5100
*200 L1=INT(L1/30)+1
*210 FOR L1=1 TO L1
*5000 NEXT L1
*5010 T1=TIM-T1
*205 T1=TIM
*5020 IF T1>=100 IF T1<200 GOTO 5050
*5030 L1=INT(L1*100/T1)+1
*5040 GOTO 205
*5050 PRINT "CZAS WYKONYWANIEA - OK. "
*5060 PRINT (T1/L1-T0/L0)*1000;"MILISEKUND"
*5100 PRINT "PODAJ INSTRUKCJE DO TESTOWANIA"
*5110 PRINT "O NR WIERSZY OD 1000 DO 4999"
*5115 PRINT "A NA KONIEC 3 GWIAZDKI '***'"
*5120 LINPUT L$
*5130 V=VAL(L$)
*5135 IF OCC(L$,"***")>0 GOTO 6050
*5140 IF PVL<=6 IF V<5000 IF V>=1000 GOTO 6000
*5145 PRINT "ZLY NUMER WIERSCA, INSTR. ZIGNOROWANA"
*5150 GOTO 5120
*6000 PRINT#2:L$
*6010 GOTO 5120
*6050 LINPUT#1:L$,L$
*6055 IF PER(1,"LINPUT")=0 GOTO 6100
*6060 LINPUT#1:L$
*6070 PRINT#2:L$
*6080 GOTO 6055
*6100
*6100RESET #2:
*6110 CHAIN "TIWRK":
*6120 END
*SAVE
*BYE

BASIC FINISH 02570
PF,INTM. 26
CO,. 26
*INTM CHR 3 3
```


BASIC

BASIC.

*RUN

RUNNING INTM 681

BASIC STOPPED 04355

*DIS TO

3

*DIS LO

176807

*OLD INTM

*RUN

RUNNING INTM 681

PODAJ INSTRUKCJE DO TESTOWANIA

O NR WIERSZY OD 1000 DO 4999

A NA KONIEC 3 GWIAZDKI '***'

?AAAAA

ZLY NUMER WIERSCA. INSTR. ZIGNOROWANA

?1000 A=2

?***

RUNNING INTM 934

NIELEGALNY NUMER KANALU LUB ARGUMENT FUNKCJI LANCUCHOWEJ--AT 60

60 FILE #2: "TIWRK"

*LIST1-120

PROG INTM

60 FILE #2: "TIWRK"

70 RESET #1:

80 RESET #2:

90 IF LO<>0 GOTO 190

100 LET LO=L1=1000

110 LET TO=TIM

120 FOR LO=1 TO LO

*OLD INTM

*NLI6050

6050 LINPUT #1: L\$,L\$

*6050 LINPUT #1:L\$,L\$,L\$

*SAVE

ZBIOR NIEDOSTEPNY (CHRONIONY)

*BYE

BASIC FINISH 02570

NP,INTM.

.

*SVE

NIEROZPOZNANA INSTRUKCJA

*SAVE

*BYE

BASIC FINISH 02570

PF,INTM. 26

CG,. 26

*INTM CHR 3 3

BASIC.

*RUN

RUNNING INTM 683

PODAJ INSTRUKCJE DO TESTOWANIA

O NR WIERSZY OD 1000 DO 4999

A NA KONIEC 3 GWIAZDKI '***'

?1000 A=2

?***

RUNNING INTM 926

PODAJ INSTRUKCJE DO TESTOWANIA

O NR WIERSZY OD 1000 DO 4999

A NA KONIEC 3 GWIAZDKI '***'

?1000 A=2-2

?1100 B=3-3

?***

RUNNING INTM 943

PODAJ INSTRUKCJE DO TESTOWANIA

O NR WIERSZY OD 1000 DO 4999

A NA KONIEC 3 GWIAZDKI '***'

?

STOP--AT 5120

*OLD INTM

*6105 T1=1

*BYE

BASIC FINISH 02570

CG, . 26

*INTM CHR 3 3

NP, .

.

*SAVE

*BYE

BASIC FINISH 02570

PF, . 26

.

*RUN

RUNNING INTM 688

PODAJ INSTRUKCJE DO TESTOWANIA

O NR WIERSZY OD 1000 DO 4999

A NA KONIEC 3 GWIAZDKI '***'

?1000 A=2

?***

RUNNING INTM 931

CZAS WYKONYWANIA - OK. .149381 MILISEKUND

PODAJ INSTRUKCJE DO TESTOWANIA

O NR WIERSZY OD 1000 DO 4999

A NA KONIEC 3 GWIAZDKI '***'

?1000 X=SIN(.01)

?***

RUNNING INTM 937

CZAS WYKONYWANIA - OK. 1.26006 MILISEKUND

PODAJ INSTRUKCJE DO TESTOWANIA

O NR WIERSZY OD 1000 DO 4999

A NA KONIEC 3 GWIAZDKI '***'

?1000 A\$=

?1000 I=ODD(I+1)

?1100 A\$=MUK\$L\$(5*(1+I), "A")

?***

RUNNING INTM 958

CZAS WYKONYWANIA - OK. 3.60477 MILISEKUND

`<instrukcja dołączania z biblioteki> ::=`
`LIBRARY <identyfikator> [, <identyfikator>]...`

Obecność takiej instrukcji w programie powoduje w chwili uruchamiania programu zleceniem RUN bądź instrukcją zmiany programu dołączenie do programu subprogramów, znajdujących się w zbiorach, których nazwy znajdują się na liście instrukcji LIBRARY. Zbiory te mogą zawierać subprogramy w postaci źródłowej bądź w postaci binarnej (w postaci obrazu pamięci) .

INSTRUKCJA ZANIEDBANIA NADMIARU UJEMNEJ CECHY

Jeżeli przy wykonywaniu programu wystąpi nadmiar ujemnej cechy, np. jeśli przy iteracji jakieś wyrażenie zmniejsza swą wartość bezwzględną, dążąc do zera, system sygnalizuje wystąpienie takiego nadmiaru wydrukiem o błędzie, poczem wstawia w miejsce wartości wyrażenia, w którym wystąpił nadmiar, zero i kontynuuje wykonywanie programu.

Instrukcja zaniedbania nadmiaru ujemnej cechy składa się ze słowa UNDERFLOW. Od chwili jej wykonania aż do zakończenia realizacji programu system nie drukuje błędu nadmiaru ujemnej cechy, a tylko przyjmuje wartość zero bez przerywania działania programu.

ASCII (ISO-7) CODES

0	C	NUL	32	040	SP	64	0100	@	96	0140	
1	01	SOH	33	041	!	65	0101	A	97	0141	a
2	02	STX	34	042	"	66	0102	B	98	0142	b
3	03	ETX	35	043	#	67	0103	C	99	0143	c
4	04	EDT	36	044	\$	68	0104	D	100	0144	d
5	05	ENQ	37	045	%	69	0105	E	101	0145	e
6	06	ACK	38	046	&	70	0106	F	102	0146	f
7	07	BEL	39	047	'	71	0107	G	103	0147	g
8	C10	BS	40	050	(	72	0110	H	104	0150	h
9	C11	HT	41	051	)	73	0111	I	105	0151	i
10	C12	LF	42	052	*	74	0112	J	106	0152	j
11	C13	VT	43	053	+	75	0113	K	107	0153	k
12	C14	FF	44	054	,	76	0114	L	108	0154	l
13	C15	CR	45	055	-	77	0115	M	109	0155	m
14	C16	SO	46	056	.	78	0116	N	110	0156	n
15	C17	SI	47	057	/	79	0117	O	111	0157	o
16	C20	DLE	48	060	0	80	0120	P	112	0160	p
17	C21	DC1	49	061	1	81	0121	Q	113	0161	q
18	C22	DC2	50	062	2	82	0122	R	114	0162	r
19	C23	DC3	51	063	3	83	0123	S	115	0163	s
20	C24	DC4	52	064	4	84	0124	T	116	0164	t
21	C25	NAK	53	065	5	85	0125	U	117	0165	u
22	C26	SYN	54	066	6	86	0126	V	118	0166	v
23	C27	ETB	55	067	7	87	0127	W	119	0167	w
24	C30	CAN	56	070	8	88	0130	X	120	0170	x
25	C31	EM	57	071	9	89	0131	Y	121	0171	y
26	C32	SUB	58	072	:	90	0132	Z	122	0172	z
27	C33	ESC	59	073	;	91	0133	[	123	0173	{
28	C34	FS	60	074	<	92	0134	\	124	0174	
29	C35	GS	61	075	=	93	0135	]	125	0175	}
30	C36	RS	62	076	>	94	0136	^	126	0176	~
31	C37	US	63	077	?	95	0137	_	127	0177	~

FUNKCJE STANDARDOWE JĘZYKA BASIC

W podanym niżej wykazie funkcji standardowych oznaczono umownie argumenty w następujący sposób:

- X, Y : dowolne wyrażenia arytmetyczne
M, N : wyrażenia wskaźnikowe indeksy tj. wyrażenia arytmetyczne, na których po obliczeniu wykonywana jest funkcja entier (największa liczba całkowita nie większa od wartości wyrażenia). Ujemne wartości wyrażen wskaźnikowych powodują wystąpienie błędów.
A\$, B\$, C\$: wyrażenia łańcuchowe

1. Podstawowe funkcje standardowe arytmetyczne

ABS(X)

Wartość bezwzględna (moduł)

ATN(X)

Wartość funkcji arcus tangens, w radianach, z przedziału $-\frac{\pi}{2}, \frac{\pi}{2}$

COS(X)

Wartość funkcji cosinus, argument w radianach. Zbyt duża wartość modułu jest traktowana jako błąd.

EXP(X)

Wartość funkcji wykładniczej e^X (e - podstawa logarytmów naturalnych).

INT(X)

Największa liczba całkowita nie większa niż X.

LOG(X)

Wartość logarytmu naturalnego X. Niedodatnie wartości argumentu powodują błąd.

RND

Kolejna liczba pseudolosowa (o rozkładzie równomiernym) z przedziału $[0,1)$.

Przy każdym uruchomieniu programu jest generowany taki sam ciąg liczb pseudolosowych, chyba, że została w programie wykonana instrukcja randomizacji.

SGN(X)

Znak (signum) wartości X

wynosi: -1 dla $X < 0$
 0 dla $X = 0$
 +1 dla $X > 0$

SIN(X)

Wartość funkcji sinus, argument w radianach. Zbyt duża wartość modułu argumentu jest traktowana jako błąd.

SQR(X)

Pierwiastek kwadratowy. Ujemna wartość argumentu jest traktowana jako błąd.

2. Inne funkcje standardowe arytmetyczne

@PI

Wartość liczby π .

ACS(X)

Wartość funkcji arcus cosinus w radianach, z przedziału $0, \pi$. Wartość modułu argumentu większa od jedności jest traktowana jako błąd.

ASN(X)

Wartość funkcji arcus sinus w radianach, z przedziału $-\frac{\pi}{2}, \frac{\pi}{2}$

Wartość argumentu większa od jedności jest traktowana jako błąd.

COT(X)

Wartość funkcji cotangens, argument w radianach.

EPS

Wartość najmniejszej liczby dodatniej, dla której są spełnione w arytmetyce maszyny nierówności

$$1 + EPS > 1$$

$$1 - EPS < 1$$

FRA(X)

Część ułamkowa wartości wyrażenia X. Za pomocą innych funkcji można ją wyrazić następująco:

$$SGN(X) * (ABS(X) - INT(ABS(X)))$$

INF

Wartość największej liczby dodatniej, dla której istnieje w arytmetyce maszyny reprezentacja +INF oraz -INF.

INX

Największa liczba, która może służyć do indeksowania wewnątrz systemu, wynosząca 32767. Stosowanie jej, wygodne w niektórych funkcjach łańcuchowych, polepsza czytelność programu. Używa się jej np. wtedy, gdy chcemy wskaźnikiem sięgnąć na pewno poza koniec wartości łańcuchowej.

MOD(X,Y)

Wartość reszty dzielenia X przez Y. Za pomocą innych funkcji można ją wyrazić następująco:

$$X - Y * INT(X/Y)$$

NIL

Wartość najmniejszej liczby dodatniej, dla której w arytmetyce maszyny istnieje reprezentacja +NIL i -NIL.

ODD(N)

Nieparzystość wyrażenia wskaźnikowego. Ma wartość 1 gdy N nieparzyste, zaś zero gdy parzyste.

TAN(X)

Wartość funkcji tangens, argument w radianach.

3. Funkcje standardowe arytmetyczne dotyczące wartości łańcuchowych

COD(N, A\$)

Wartość kodu ISO-7 N-tego znaku wartości łańcuchowej A\$.

LEN(A\$)

Długość (liczba znaków) wartości łańcuchowej A\$.

LOL

Długość w znakach wartości nadanej zmiennej wynikowej w ostatnio wykonanym przypisaniu (podstawieniu) łańcuchowym. Nie ma określonej wartości po wejściu lub powrocie z subprogramu oraz po przejściu i powrocie z obliczania wartości funkcji definiowanej.

LON

Długość w znakach najdłuższej wartości łańcuchowej, która dałaby się w danej chwili utworzyć. Stanowi długość dostępnej programowi w danej chwili pamięci (nie licząc samego programu, obszaru zmiennych i stosów) w bajtach.

OCC(A\$, B\$)

Liczba wystąpień B\$ w A\$. Ma wartość zero, gdy B\$ w A\$ nie występuje ni razu.

POL(A\$, B\$)

Położenie (kolejny numer znaku) pierwszego znaku ostatniego wystąpienia B\$ w A\$. Ma wartość zero, gdy B\$ w A\$ nie występuje.

POS(A\$, B\$)

Położenie pierwszego znaku pierwszego wystąpienia B\$ w A\$. Ma wartość zero, gdy B\$ w A\$ nie występuje.

PSL

Wartość ostatnio obliczonej funkcji $POS(A\$,B\$)$, $POL(A\$,B\$)$ lub $PSN(A\$,B\$,N)$.

$PSN(A\$,B\$,N)$

Położenie pierwszego znaku N-tego wystąpienia $B\$$ w $A\$$. Ma wartość zero, jeśli $B\$$ występuje w $A\$$ mniej niż N razy.

PVL

Funkcja ta "współpracuje" z funkcją $VAL(A\$)$. Wartością jej jest pozycja pierwszego znaku w wartości $A\$$ za liczbą "pobrąną" ostatnio wykonaną funkcją $VAL(A\$)$ lub drugiej spacji, jeśli liczba kończy się spacjami. Pojedyncze spacje wewnątrz liczby są ignorowane. Jeśli argument ostatnio wykonanej funkcji $VAL(A\$)$ nie zawierał znaków tworzących liczbę (tzn. ani jednej cyfry), wartość PVL wynosi zero. Wartość początkowa: zero.

$VAL(A\$)$

Ma taką wartość, jaką miałaby zmienna arytmetyczna V, gdyby ciąg znaków tworzących wartość łańcuchową $A\$$ podano na wejście przy wykonywaniu instrukcji INPUT V.

4. Funkcje standardowe arytmetyczne dotyczące zmiennych indeksowanych

COL

Liczba kolumn (wartość graniczna drugiego subskryptu) zmiennej indeksowanej, na którą ostatnio dokonano podstawienia instrukcją przypisania zmiennych indeksowanych MAT. Ma wartość 1, jeśli ostatnio wykonana instrukcja MAT dotyczyła wektora. Wartość początkowa: zero.

DET

Wyznacznik ostatnio odwracanej macierzy. Osobliwość macierzy nie jest przy odwracaniu sygnalizowana jako błąd. Wartość początkowa: zero.

ROW

Liczba wierszy (wartość graniczna pierwszego subskryptu) zmiennej indeksowanej (macierzy lub wektora), na którą ostatnio dokonano podstawienia instrukcją MAT. Wartość początkowa: zero.

5. Funkcje standardowe arytmetyczne systemowe

DAY

Liczba porządkowa bieżącego dnia w miesiącu.

DYR

Liczba porządkowa bieżącego dnia w roku.

LOC(N)

Aktualna wartość wskaźnika zbioru o dostępie swobodnym na kanale N. Wskazuje element zbioru, którego będzie dotyczyła następna operacja czytania bądź zapisu dla tego kanału (pierwszy element zbioru ma wskaźnik zero).

LOF(N)

Długość (liczba elementów) zbioru o dostępie swobodnym na kanale N.

LOP(N)

Położenie (numer pozycji) w wierszu znaku, którego będzie dotyczyła następna operacja wprowadzania lub drukowania dla zbioru znakowego lub urządzenia na kanale N. Najbardziej skrajny znak z lewej strony ma numer pozycji zero.

MON

Numer porządkowy bieżącego miesiąca w roku.

MRG(N)

Wartość marginesu zbioru na kanale N.

PER(N, A\$)

Funkcja, pozwalająca stwierdzić wykonalność w danym momencie operacji na kanale N i uniknąć w ten sposób zatrzymania programu. Rodzaj operacji określony jest wartością A\$. Jeśli operacja jest w danym momencie dozwolona, funkcja ma wartość 1, jeśli operacja jest niedozwolona - wartość zero. Nielegalna wartość A\$ daje wartość funkcji równą -1. Legalne wartości drugiego argumentu A\$ i operacje, dla których wartość funkcji PER(N, A\$) wynosi 1, zestawione są poniżej.

- "FILE" : można przypisać zbiór do kanału N instrukcją FILE tzn. kanał N jest nieaktywny.
- "SCRATCH" : zbiór na kanale N można wymazać instrukcją SCRATCH.
- "MARGIN" : wolno zmienić wartość marginesu zbioru na kanale N - tzn. zbiór na kanale N jest zbiorem znakowym lub pustym zbiorem o dostępie swobodnym.
- "PRINT" : wolno wykonać dla kanału N operację drukowania.
- "INPUT" : można wprowadzić informację z kanału N - zbiór na tym kanale jest zbiorem znakowym i jego zawartość może być badana.
- "LINPUT" : jak "INPUT".
- "WRITE" : do zbioru na kanale N można zapisać wartość arytmetyczną.
- "WRITE\$" : do zbioru na kanale N można zapisać wartość łańcuchową.
- "READ" : zbiór na kanale N jest zbiorem arytmetycznym o dostępie swobodnym i można badać jego zawartość.

"READ\$" : zbiór na kanale N jest zbiorem łańcuchowym o dostępie swobodnym i można badać jego zawartość.

SEC

Numer porządkowy bieżącej sekundy doby, tj. liczba sekund, jaka upłynęła od północy do danej chwili.

TIM

Liczba sekund, jaka upłynęła od momentu uruchomienia programu do danej chwili.

YEA

Numer porządkowy roku, np. 1981.

6. Funkcje standardowe łańcuchowe

CHR\$(N)

Wartość łańcuchowa znaku o kodzie ISO-7 równym N modulo 128. Kod o wartości zero daje łańcuch pusty.

DEL\$(A\$,B\$)

Wartość A\$ z usuniętym, jeśli było, pierwszym wystąpieniem B\$.

GAP\$(N)

Wartość łańcuchowa utworzona z N znaków spacji SP.

LIN\$(N)

Wartość łańcuchowa utworzona ze znaku powrotu karetki CR i N znaków zmiany wiersza LF.

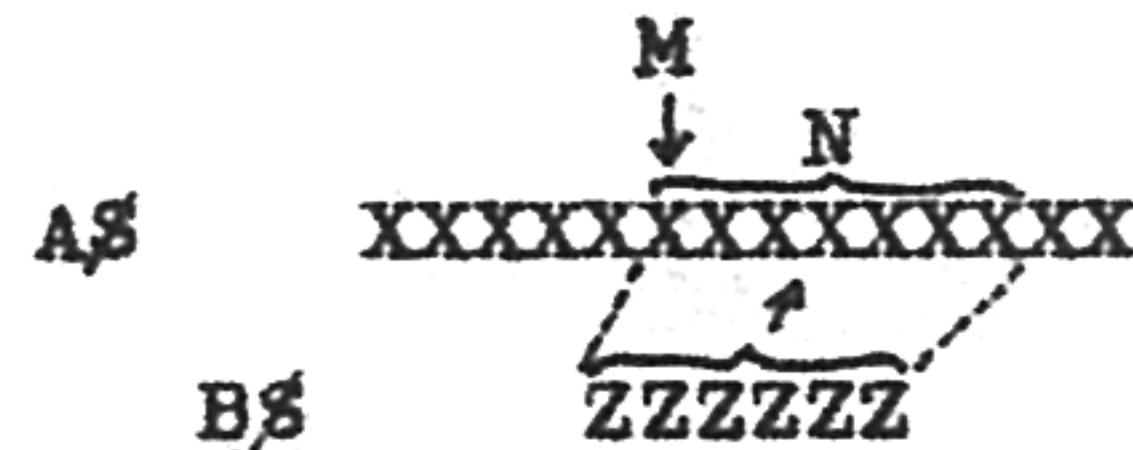
MER\$(A\$,B\$,M,N)

Wartość A\$, z której usunięto N znaków od M-tego począwszy, a na ich miejsce wstawiono wartość B\$.

Wartość tej funkcji można przedstawić na pomocą innych funkcji następująco:

$SEG\$(A\$,1,M-1) \& B\$ \& SEG\$(A\$,M+N,INX)$

zaś schematycznie



Poniżej podano wartości funkcji dla szczególnych wartości M i N

jeśli $N = \emptyset$:

$SEG\$(A\$,1,M-1) \& B\$ \& SEG\$(A\$,M,INX)$

jeśli $M = \emptyset$:

$B\$ \& SEG\$(A\$,N+1,INX)$

jeśli $M > LEN(A\$) + 1$:

$B\$$

jeśli $M+N > LEN(A\$)+1$:

$SEG\$(A\$,1,M-1) \& B\$$

$MUL\$(N,A\$)$

Wartość łańcuchowa utworzona z N wartości $A\$$.

Np. $MUL\$(3,"ABC")$ wynosi "ABCABCABC".

$REP\$(A\$,B\$,C\$)$

Wartość łańcuchowa utworzona przez zastąpienie w $A\$$ pierwszego wystąpienia $B\$$, jeśli istniało, przez $C\$$.

$SDL\$(A\$,B\$)$

Wartość $A\$$, z której usunięto wszystkie wystąpienia $B\$$.

$SEG\$(A\$,M,N)$

Segment $A\$$ od M -tego do N -tego znaku włącznie.

Jeśli N jest większe od $LEN(A\$)$, to przyjmowane

jest jako $LEN(A\$)$. Wartości $M > LEN(A\$)$ lub $N < M$ dają

jako wartość funkcji łańcuch pusty. Wartości $M \leq 0$

lub $N \leq 0$ są błędem.

SGN $\$$ (X)

Wartość łańcuchowa znaku (signum) wyrażenia arytmetycznego X. Ma wartość:

"-" dla $X < 0$
" " (spacja) dla $X = 0$
"+" dla $X > 0$

SRP $\$$ (A $\$$, B $\$$, C $\$$)

Wartość A $\$$, w której wszystkie wystąpienia B $\$$ zostały zastąpione przez C $\$$.

STR $\$$ (X)

Wartość łańcuchowa, utworzona ze znaków, które zostałyby wyprowadzone w wyniku wykonania instrukcji:

PRINT X;

Przykładowo, wartością funkcji STR $\$$ (-17.3300) będzie "-17.33".

STF(X, A $\$$)

Wartość łańcuchowa, utworzona ze znaków, które zostałyby wyprowadzone w wyniku wykonania instrukcji:

PRINT USING A $\$$: X;

Wartość A $\$$ musi być poprawnie skonstruowanym polem arytmetycznym, zawierającym jedynie znaki
+ - . # ↑

SUB $\$$ (A $\$$, M, N)

Wartość łańcuchowa utworzona z N kolejnych znaków A $\$$, począwszy od M-tego znaku.

Wartości $M > \text{LEN}(A\mathbb{S})$ lub $N = 0$ dają jako wartość funkcji łańcuch pusty. Jeśli N jest większe niż $\text{LEN}(A\mathbb{S}) - M + 1$, przyjmowane jest jako równe $\text{LEN}(A\mathbb{S}) - M + 1$.

Wartości $M \leq 0$ lub $N < 0$ są błędem.

7. Funkcje standardowe łańcuchowe systemowe

CLK\$

Ośmioznakowy łańcuch podający bieżący czas w godzinach, minutach i sekundach, np. "17:03:31".

CUF\$

Łańcuch podający nazwę zbioru lub podzbioru roboczego (programu głównego lub subprogramu), w którym wystąpiłowołanie funkcji.

DAT\$

Dziewięcioznakowy łańcuch, podający datę np. "20 JUN 80".

JOB\$

Łańcuch, podający nazwę zbioru roboczego (programu głównego).

SY\$

Łańcuch podający nazwę systemu.

USR\$

Łańcuch podający nazwę użytkownika.

KONWERSACJA UŻYTKOWNIKA Z SYSTEMEM BASIC

BASIC jest systemem konwersacyjnym. Praca w tym systemie polega na tym, że użytkownik wprowadza wiersz informacji, np. wiersz zlecenia, a system zlecenie wykonuje i wyprowadza informację żadaną przez użytkownika lub potwierdzenie - po czym oczekuje na następny wiersz informacji od użytkownika.

Ze względu na złożoność systemu praca ta może odbywać się w kilku kontekstach, w zależności od tego, z jaką częścią systemu BASIC użytkownik współpracuje.

Dla umożliwienia szybkiej orientacji co do kontekstu, system zgłasza nasłuch czyli oczekiwanie na informację od użytkownika wyprowadzeniem na końcówkę jednego znaku w nowym wierszu. Znak ten jest inny dla każdego z kontekstów, a jego wyprowadzenie stanowi jednocześnie potwierdzenie realizacji zleceń. Poniżej podane są te znaki i odpowiadające im konteksty.

- * gwiazdka - kontekst zbioru roboczego (programu głównego).
- znak numeru - kontekst podzbioru roboczego (subprogramu).
- ? znak zapytania - uruchomiony program użytkownika oczekuje na wprowadzenie informacji z końcówki w trakcie wykonywania instrukcji INPUT, LINPUT, MAT INPUT bądź MAT LINPUT.
- ! wykrzyknik - przy wprowadzaniu programu, w trybie programowym, nastąpiło przekroczenie zakresu pamięci - pojemność pamięci dostępnej użytkownikowi okazała się zbyt mała. Program zawarty w zbiorze jest niekompletny i należy go usunąć, by można było w pełni prawidłowo pracować w trybie programowym - należy np. wydać zlecenie OLD lub NEW. System przyjmuje w tym kontekście zlecenia trybu programowego.

Brak jakiegokolwiek z powyższych znaków w nowym wierszu może oznaczać 3 sytuacje:

1. Wykonuje się właśnie program użytkownika uruchomiony zleceniem RUN i program ten nie współpracuje z końcówką, bądź realizuje się przyjęte przez system BASIC zlecenie.
2. System BASIC jest uszkodzony i nie pracuje.
3. Kończówka jest na nasłuchu systemu operacyjnego CROOK.

Jeżeli system BASIC jest na nasłuchu, to wprowadzenie pustego wiersza, tj. znaku nowego wiersza LF powinno spowodować wyprowadzenie znaku kontekstu w nowym wierszu (poprzedzonego niekiedy komunikatem o błędzie). Jeżeli maszyna sama wyprowadzi dodatkowy nowy wiersz, wtedy mamy do czynienia z nasłuchem systemu operacyjnego. Wprowadzenie w takiej sytuacji znaku DC4 (CNTRL + H) powinno spowodować przejście do nowego wiersza.

Zasady wprowadzania informacji z końcówki są podobne jak w innych programach pracujących pod systemem operacyjnym CROOK.

Krótko można je zreasumować następująco:

- informację wprowadza się wierszami. Znakiem kończącym wiersz jest znak zmiany wiersza LF. Do chwili wprowadzenia tego znaku informacja jest przyjmowana przez system operacyjny, a nie przez BASIC. Dopiero po wprowadzeniu znaku LF cały wiersz jest przekazywany systemowi BASIC.
- wprowadzenie n znaków BS kasuje n ostatnio wprowadzonych w tym wierszu znaków. Generuje się je w taki sposób, że trzymając naciśnięty klawisz CNTRL naciska się n razy klawisz H. Przy pracy na monitorze ekranowym powoduje to cofnięcie kursora o n pozycji znakowych.
- wprowadzenie znaku CAN (naciśnięcie klawisza CNTRL i jednocześnie klawisza X) powoduje skasowanie całej informacji dotychczas wprowadzonej w bieżącym wierszu.

Uruchomienie systemu BASIC

System BASIC uruchamia się, wprowadzając w trybie nasłuchu systemu operacyjnego CROOK zlecenie

BASIC.

W wyniku wykonania tego zlecenia system BASIC zgłosi się na końcówce, wyprowadzając znak nasłuchu trybu programowego, tj. gwiazdkę * lub znak numeru #.

Przejście do systemu operacyjnego można uzyskać bądź przez wygenerowanie przerwania (naciśnięcie klawisza BREAK bądź, w niektórych wersjach systemu, klawisza oznaczonego OPRQ na płycie czołowej maszyny przy jednoczesnym ustawieniu na klawiszach informacyjnych numeru przypisanego końcówce).

ZLECENIA JĘZYKA BASIC

Praca w trybie programowym ma na celu tworzenie, poprawianie i wykonywanie programów. Tworzenie i poprawianie programów polega na wprowadzeniu z końcówki wierszy programu - tj. wierszy rozpoczynających się numerem wiersza, zawierających instrukcje, deklaracje itp. języka BASIC.

Wiersze te są przez program BASIC sprawdzane pod kątem poprawności syntaktycznej i - jeśli są poprawne - umieszczane w zbiorze roboczym, stając się częścią tworzonego programu. Zbiór roboczy jest to blok pamięci operacyjnej, kopiowany i ściągany w miarę potrzeby na dysk i z dysku przez system BASIC w sposób niewidoczny dla użytkownika.

W zbiorze roboczym mogą się zawierać podzbiory robocze, zawierające subprogramy (procedury). Pracując w trybie programowym użytkownik jest bądź w kontekście programu głównego (zbioru roboczego), którego znakiem nasłuchu jest gwiazdka *, bądź w kontekście subprogramu (podzbioru roboczego), którego znakiem nasłuchu jest znak numeru #.

Wiersze nie opatrzone numerem, wprowadzane w trybie programowym są przez system traktowane jako zlecenia języka BASIC. Poniżej opisano te zlecenia i wykonywane przez nie czynności.

Niektóre kody zleceń mogą mieć postać pełną i skróconą. Można ich używać wymiennie.

Podobnie jak w wierszach programów języka BASIC, tak i w zleceniach spacje nie mają znaczenia i są przez system pomijane.

Nie wolno jednak umieszczać więcej niż jednej spacji wewnątrz znaków tworzących liczbę oraz żadnych spacji w identyfikatorach (np. nazwach zbiorów).

NEW [$\langle$ identyfikator $\rangle$]

$\langle$ identyfikator $\rangle ::= \langle$ litera $\rangle \left[\begin{array}{l} \langle$ litera $\rangle \\ \langle$ cyfra $\rangle \end{array} \right] \dots$

System uwzględnia tylko sześć pierwszych znaków identyfikatora - dalsze są ignorowane.

Zlecenie podane z parametrem powoduje inicjalizację zbioru roboczego: zawartość zbioru roboczego i podzbiorów roboczych (programy i subprogramy) jest usuwana. Zbiorowi roboczemu nadawana jest nazwa podana jako parametr. Po wykonaniu zlecenia system zgłasza się znakiem gwiazdki * - nasłuchem zbioru roboczego programu głównego.

Wydanie zlecenia bez parametru powoduje usunięcie tego tylko zbioru lub podzbioru roboczego, w którego kontekście wydano zlecenie i przejście do kontekstu zbioru roboczego - programu głównego. Zaleca się jednak w takich przypadkach używać raczej zlecenia UNS.

PROG $\langle$ identyfikator $\rangle$

PROGrama

Nadaje zbiorowi roboczemu - programowi głównemu, nazwę podaną jako parametr, usuwając starą nazwę. Wydane w kontekście subprogramu powoduje przejście do kontekstu programu głównego.

NAME $\langle$ identyfikator $\rangle$

Powoduje przejście do kontekstu podzbioru roboczego (subprogramu) o nazwie podanej jako parametr. Jeśli podzbioru o takiej nazwie nie ma, zostaje on utworzony, co system potwierdza komunikatem:

NEW : $\langle$ identyfikator $\rangle$

i zgłasza się nasłuchem w tym podzbiorze.

Powrót do kontekstu programu głównego można uzyskać wprowadzając znak DC4.

UNS <identyfikator>

UNSave

Zlecenie powoduje usunięcie podzbioru lub zbioru roboczego o podanej nazwie i przejście do kontekstu programu głównego.

CAT [# <nr we/wy>]

CATalogue

<nr we/wy> ::= <liczba naturalna mniejsza od 8>

Zlecenie powoduje wydruk nazw zbioru roboczego i podzbiorów roboczych na końcówce lub wskazanym strumieniu.

Drukują się: nazwa zbioru programu głównego oraz nazwy podzbiorów subprogramów z treścią pierwszego wiersza każdego subprogramu.

Dla programów zawierających subprogramy binarne drukowany jest jeszcze nagłówek BIN i wiersze nagłówków subprogramów, w których numer jest adresem startowym subprogramu.

Po wykonaniu zlecenia system przechodzi do nasłuchu w kontekście programu głównego.

MERGE { # <nr we/wy>
 <nazwa zbioru> }

<nazwa zbioru> ::= <identyfikator>

Zlecenie powoduje pobieranie informacji ze wskazanego wejścia i traktowanie jej tak, jakby była wprowadzana z końcówki w trybie programowym, tzn. np. wprowadzenie jako wiersz programu wiersza błędnego spowoduje wydruk komunikatu o błędzie, wprowadzenie napisu NAME <identyfikator> spowoduje zmianę kontekstu na kontekst subprogramu o odpowiedniej nazwie (p. wyżej zlecenie NAME) itd. Wprowadzanie informacji kończy się z chwilą napotkania na wejściu znacznika końca zbioru, po czym system przechodzi na nasłuch w kontekście programu głównego.

SAVE [<identyfikator> [, <identyfikator>] ...]

Wydanie zlecenia bez parametrów spowoduje wpisanie całości zbioru roboczego - programu głównego i subprogramów w postaci źródłowej, z nagłówkami - do zbioru o identyfikatorze takim jak nazwa programu głównego.

Jeśli zbiór o takiej nazwie i odpowiedniego typu (CHR) już istnieje i nie jest chroniony przed zapisem - treść kompletnego programu zostanie doń wpisana ze zniszczeniem poprzedniej zawartości. Jeśli nie istnieje - zostanie utworzony zbiór typu CHR i umieszczona w nim treść całości programu.

Jeśli zbiór istnieje, lecz jest chroniony przed zapisem, zlecenie nie zostanie wykonane i wydrukuje się komunikat o błędzie. Zlecenie nie powoduje wyprowadzenia subprogramów binarnych dołączonych do zbioru roboczego.

Wydanie zlecenia z parametrami spowoduje wyprowadzenie tych jednostek programu, których identyfikatory podano jako parametry. Jako nazwa zbioru, do którego nastąpi wyprowadzenie zostanie wzięty identyfikator, który jest pierwszym parametrem.

OLD <nazwa zbioru>

Zlecenie powoduje usunięcie całości programu znajdującego się w zbiorze roboczym i podzbiorach roboczych, a następnie wprowadzenie na jego miejsce programu zawartego w zbiorze o nazwie określonej parametrem. Zbiór musi zawierać program w postaci źródłowej. Zlecenie to służy przede wszystkim do umieszczania w pamięci (w zbiorze roboczym i podzbiorach roboczych) programów, zachowanych uprzednio w zbiorach zleceniem SAVE.

NSAVE <identyfikator>

No name SAVE

Zlecenie powoduje wpisanie treści programu głównego lub subprogramu o nazwie określonej parametrem, w postaci źródłowej, lecz bez nagłówka NAME lub PROG do zbioru o nazwie takiej jak parametr oraz przejście do kontekstu programu głównego.

$$\text{LIS}[T] \left[\begin{array}{l} \# \langle \text{nr we/wy} \rangle \\ [\# \langle \text{nr we/wy} \rangle,] \langle \text{lista numerów wierszy} \rangle \end{array} \right]$$
$$\langle \text{lista numerów wierszy} \rangle ::= \left\{ \begin{array}{l} - \langle \text{nr wiersza} \rangle \\ \langle \text{nr wiersza} \rangle [- \langle \text{nr wiersza} \rangle] \end{array} \right\}$$
$$[, \langle \text{nr wiersza} \rangle [- \langle \text{nr wiersza} \rangle]] \dots [, \langle \text{nr wiersza} \rangle -]$$

Wydanie zlecenia bez listy numerów wierszy powoduje wyprowadzenie zawartości całego zbioru lub podzbioru roboczego, w którego kontekście zostało wydane. Pozwala zatem w każdej chwili sprawdzić, jak dokładnie wygląda tworzony właśnie z końcówki program lub subprogram.

Podanie listy numerów wierszy określa fragmenty programu lub subprogramu, które chcemy wyprowadzić.

I tak:

- $\langle \text{nr wiersza} \rangle$ oznacza wiersze począwszy od wiersza o najniższym numerze do wiersza, którego numer podano, włącznie.

$\langle \text{nr wiersza} \rangle - \langle \text{nr wiersza} \rangle$ oznacza wszystkie wiersze zawarte w podanym zakresie, z wierszami o podanych numerach włącznie.

$\langle \text{nr wiersza} \rangle -$ oznacza wszystkie wiersze od podanego numeru do końca do wiersza o największym numerze.

Numer wierszy na liście muszą być umieszczone w porządku wzrastającym. Jeżeli numer wiersza podany na liście nie występuje w programie, brany jest najbliższy większy numer, albo najwyższy numer, jeżeli numer podany jako parametr jest od niego większy.

Przykładowo, zlecenie

LIST -200, 300, 450-600

spowoduje wyprowadzenie wszystkich wierszy o numerach mniejszych od 200, wiersza 200, wiersza 300 oraz wierszy o numerach od 450 do 600 włącznie.

Podanie pierwszego parametru powoduje wyprowadzenie nie na końcówkę, a na wskazane urządzenie (strumień we/wy).

PLI [$\#$ <nr we/wy>]

Program Listing

Wykonanie zlecenia powoduje wydrukowanie lub wyprowadzenie całości zbioru roboczego i podzbiorów roboczych (programu głównego i subprogramów), z nagłówkami, w takiej kolejności w jakiej nagłówki są drukowane przy wykonywaniu zlecenia CAT, oraz przejście do kontekstu programu głównego.

RUN

Zlecenie powoduje uruchomienie programu zawartego w zbiorze roboczym (w pamięci). Najpierw wykonywane jest sprawdzenie programu jako całości. A więc sprawdza się np. czy wszystkie wiersze, do których są w programie skoki, istnieją; czy pętle programu są właściwie skonstruowane; czy istnieją subprogramy, których występują wołania; czy są odpowiednie definicje funkcji itp.

Jeżeli w programie lub subprogramach występuje jakikolwiek błąd - drukuje się odpowiedni komunikat. Jeżeli sprawdzenie wykazało pełną poprawność programu - system rozpoczyna jego wykonywanie od najniższego numeru wiersza programu głównego.

W czasie wykonywania programu użytkownik ma możliwość przerwania go i przejścia na nasłuch systemu operacyjnego, generując przerwanie.

Oczekiwanie programu na wprowadzenie danych z końcówki np. instrukcjami INPUT lub LINPUT sygnalizowane jest znakiem zapytania na końcówce (chyba, że przed taką instrukcją wykonała się instrukcja drukowania na końcówkę zakończona średnikiem).

DIS $\left\{ \begin{array}{l} \langle \text{zmienna arytmetyczna prosta} \rangle \\ \langle \text{funkcja definiowana arytmetyczna} \rangle \end{array} \right\}$ DISplay

Zlecenia tego używa się po zakończeniu wykonywania bądź po przerwaniu pracy programu. Powoduje ono wydrukowanie wartości zmiennej lub funkcji definiowanej podanej jako parametr. Wartość ta jest taka, jaką zmienna lub funkcja miała w chwili zatrzymania lub zakończenia programu. Dla sprawdzenia

wartości zmiennej subprogramu należy uprzednio przejść do kontekstu podzbioru roboczego zawierającego ten subprogram - zleceniem NAME.

Zlecenie DIS pozwala również stwierdzić, czy zmienna o podanej nazwie występuje w programie, co jest pomocne wtedy, gdy chcemy wprowadzić do programu nową zmienną, a nie pamiętamy, czyśmy jej już w programie nie użyli. Wydanie zlecenia dla istniejącej zmiennej, ale przed uruchomieniem programu może spowodować wystąpienie błędu wynikającego z próby pobrania przez system z pamięci liczby nieznormalizowanej.

Jeżeli zmienna o podanej nazwie nie występuje w danej jednostce programowej - system wydrukuje odpowiedni komunikat.

REJ [ECT] <lista numerów wierszy>

Zlecenie powoduje usunięcie z programu lub subprogramu, w którego kontekście jest wydane, wierszy o numerach określonych listą. Lista numerów wierszy jest interpretowana tak jak zlecenie LIST, z tym, że:

- jeżeli na liście parametrów występuje pojedynczy numer a wiersza o takim numerze nie ma w programie lub subprogramie, to taki parametr jest ignorowany;
- jeżeli w programie nie ma wiersza o takim numerze jak parametr będący górną granicą w parze numerów, to ostatnim usuniętym na podstawie takiej pary wierszem będzie wiersz programu o numerze bezpośrednio niższym;
- podobnie. jeżeli w programie nie ma wiersza o takim numerze jak parametr będący dolną granicą w parze numerów, to pierwszym wierszem usuniętym na podstawie takiej pary będzie wiersz programu o numerze bezpośrednio wyższym.

RET [AIN] <lista numerów wierszy>

Zlecenie powoduje zachowanie w programie lub subprogramie, w którego kontekście zostało wydane, wierszy o numerach określonych listą, a usunięcie wszystkich pozostałych

wierszy. Lista numerów wierszy jest interpretowana tak, jak w zleceniu REJECT powyżej.

RES[EQUENCE][<nr wiersza>][,<nr wiersza>[,<nr wiersza>
[,<nr wiersza>]]]

Zlecenie służy do przenumrowania wierszy programu lub subprogramu, w którego kontekście zostało wydane. Znaczenie czterech kolejnych parametrów jest następujące:

1. parametr: stary pierwszy numer wiersza, od którego ma się zacząć przenumrowanie. Wiersze o niższych numerach nie będą przenumrowywane;
2. parametr: nowy pierwszy numer wiersza, który zostanie nadany wierszowi określonym pierwszemu parametrem;
3. parametr: przyrost numerów wierszy;
4. parametr: numer ostatniego wiersza (starego), na którym ma się zakończyć przenumrowywanie.

Pominięcie parametrów powoduje przyjęcie przez system: najmniejszego z istniejących numerów wierszy jako pierwszego parametru, 100 jako drugiego, 10 jako trzeciego i największego z istniejących numerów wierszy jako czwartego.

Należy zwrócić uwagę, że generowane przez zlecenie numery wierszy mogą przy niewłaściwym doborze parametrów być takie same jak już istniejące i nie podlegające przenumrowaniu numery wierszy programu. Wtedy stare wiersze zostaną zastąpione nowymi.

Zlecenie przenumrowuje również we właściwy sposób numery wierszy znajdujące się w instrukcjach warunkowych i skokach, tak, by nie zmienił się sposób wykonywania programu.

Przed wykonaniem zlecenia sprawdzana jest poprawność programu ze względu na realizację skoków. Jeśli wiersz, do którego jest w programie skok nie istnieje - drukuje się komunikat o błędzie i przenumrowanie nie jest wykonywane.

DMP <identyfikator>

DuMP

Zlecenie powoduje utworzenie obrazu pamięci programu wraz z wszystkimi subprogramami, również binarnymi, w zbiorze o nazwie podanej parametrem i przejście do kontekstu programu głównego. Zbiór zawierający obraz pamięci programu ma oznacznik typu BAE.

VRU <identyfikator>

VeRify and dUmp

Zlecenie działa tak jak zlecenie DMP, jednak przed przeniesieniem obrazu pamięci do zbioru wykonywana jest kontrola poprawności programu jako całości, podobnie jak przy realizacji zlecenia RUN. Jeśli w programie jest błąd, drukuje się komunikat o błędzie i przeniesienie obrazu pamięci do zbioru nie jest wykonywane.

Obraz pamięci programu, utworzony zleceniem VRU ma taką postać, że nadaje się bezpośrednio do wykonywania, bez żadnych dalszych zabiegów. Ściągnięcie i wykonywanie takiego programu uzyskamy wydając zlecenie LGO. Dzięki temu można zaoszczędzić czas potrzebny na fazę sprawdzania i weryfikacji programu, występującą przy wykonywaniu zlecenia RUN. Przy dużych programach czynności te mogą zajmować maszynie do kilku minut.

LGO <identyfikator>

Load and GO

Zlecenie służy do ściągania do pamięci programu uprzednio przeniesionego do zbioru zleceniem DMP lub VRU. Jeżeli zbiór został utworzony zleceniem VRU - program zostanie po ściągnięciu od razu uruchomiony.

Tak ściągnięty program można w normalny sposób poprawiać, modyfikować itp.

REN <identyfikator> , <identyfikator>

REName

Zlecenie służy do zmiany nazwy subprogramu z podanej pierwszym parametrem na podaną drugim parametrem. Zmienia-
na jest nazwa podzbioru, nazwa w nagłówku subprogramu po napisie SUB oraz wszystkie wystąpienia tej nazwy w woła-

łach subprogramu po napisie CALL, we wszystkich jednostkach programowych znajdujących się w pamięci.

Po wykonaniu zlecenia system zgłasza się w kontekście programu głównego.

APP <identyfikator> [, <identyfikator>]... APPend

Zlecenie pozwala dołączyć do programu subprogramy napisane w języku maszynowym. W celu znalezienia subprogramów przeszukiwane są zbiory bibliotekarza i zbiory użytkownika.

BYE

Wydanie tego zlecenia powoduje zakończenie sesji w systemie BASIC i przejście na nasłuch systemu operacyjnego CROOK.

COM <identyfikator>

COMpile

Zlecenie powoduje kompilację subprogramu o nazwie podanej parametrem, oraz wszystkich subprogramów z niego wołanych. Po wykonaniu zlecenia subprogram w postaci binarnej znajduje się w zbiorze typu ABS o nazwie takiej jak parametr zlecenia, jak również dołączony jest w postaci binarnej do systemu BASIC, co umożliwia uruchomienie programu zleceniem RUN.

Na subprogramy, które mogą podlegać kompilacji, nałożone są pewne ograniczenia. Oto one:

- subprogram nie może zawierać instrukcji wejścia/wyjścia: INPUT, PRINT, READ, WRITE, FILE (czytanie z programu, z bloku DATA, może wystąpić).
- mogą występować tylko operacje arytmetyczne, operacji i zmiennych łańcuchowych kompilator nie zna.
- nie mogą występować operacje na zmiennych indeksowanych MAT.
- jako parametr subprogramu nie może występować funkcja definiowana. Funkcje definiowane mogą jednakże występować jako parametry subprogramów wołanych z subprogramów podlegających kompilacji.
- z subprogramów podlegających kompilacji nie może być wołany subprogram binarny.

Przy wykonywaniu zlecenia system tworzy jeszcze następujące zbiory:

1. Zbiór o nazwie @@FB zawiera program główny i subprogramy nie podlegające kompilacji, uzupełnione o zlecenie APP, które przy ściąganiu tego zbioru do pamięci zleceniem OLD lub MERGE spowoduje dołączenie skompilowanego subprogramu (subprogramów) do systemu.
2. Zbiór o nazwie utworzonej z liter CH oraz czterech pierwszych znaków identyfikatora kompilowanego subprogramu zawiera pełny tekst kompilowanego subprogramu (subprogramów) w języku symbolicznym assemblerze ASSM.

Tekst tego drugiego zbioru może służyć jako wersja wyjściowa do pisania subprogramów binarnych, drogą wprowadzania w nim zmian i ulepszeń.

Dla uzyskania z takiego uzupełnionego kodu w języku symbolicznym ASSM subprogramu binarnego należy tekst źródłowy dwukrotnie stranslować specjalną wersją assemblera - ASSMB. Tak otrzymany program binarny można dołączać do systemu BASIC zleceniem APP.

WSPÓŁPRACA Z SYSTEMEM OPERACYJNYM CROOK-3

Programy zarządzające systemem zbiorów wchodzi w skład systemu operacyjnego CROOK-3. System ten zapewnia również ochronę zbiorów, ochronę dostępu do systemu oraz nadzoruje wykonywanie programów - z których jednym jest zespół programów tworzących system BASIC. Dla wygody użytkowników systemu BASIC podano w niniejszej części podręcznika niektóre wiadomości o własnościach i o posługiwaniu się systemem CROOK, w zakresie niezbędnym do korzystania z systemem BASIC.

Uruchomienie maszyny i systemu

Przy wznowieniu pracy system drukuje na końcu komunikat

SET TIME

Należy wtedy wprowadzić bieżący czas w postaci

$\langle \text{godzina} \rangle [, \langle \text{minuty} \rangle [, \langle \text{sekundy} \rangle]]$.

sprawdziwszy, czy jest załączony zegar maszyny (klucz CLK). System po przyjęciu czasu poda datę. Jeśli data jest prawidłowa, należy to potwierdzić wprowadzeniem kropki lub znaku LF, co kończy fazę ustalania daty i czasu. Jeżeli data nie jest prawidłowa, należy wprowadzić liczbę naturalną poprzedzoną znakiem, wskazującą systemowi, o ile dni ma zmienić datę i zakończyć ją kropką lub znakiem LF. System wydrukuje wówczas poprawioną datę, z którą należy postąpić jak powyżej - tzn. zaakceptować lub poprawić.

Po zakończeniu tej fazy system zgłasza się komunikatem

WHO ARE YOU ?

oznaczającym, że użytkownik może się wpisać do systemu.

Czasami zdarza się, że wskutek zakłóceń lub innych trudnych do określenia przyczyn system sprawia wrażenie, jakby nie działał. Należy wtedy sprawdzić stan rejestrów SR oraz IR wybierając przełącznikiem odpowiedni rejestr i odczytując

jego stan z 16 lampek informacyjnych -lampka zapalona odpowiada bitowi równemu 1 .

Przy poprawnie działającym systemie bity 0...9 rejestru SR powinny być równe 1, zaś w rejestrze IR bity 0...2 oraz 4 i 5 powinny być równe 1, zaś bity 6...16 równe zero.

Wówczas należy sprawdzić, czy dysk jest załączony i działający oraz czy końcówki są sprawne (czy są włączone ON-LINE i na transmisję duplex, czy jest w nich papier itp.).

Jeżeli system nie działa, tzn. któryś z bitów 0...9 rejestru SR jest równy zero, albo któryś z bitów 10...15 rejestru IR jest równy 1, a mamy podstawy przypuszczać, że nie nastąpiło zniszczenie zawartości pamięci systemu, należy go uruchomić w następujący sposób:

1. Zatrzymać maszynę - ustawić klucz START na pulpicie w dolne położenie
2. Wyzerować maszynę - nacisnąć na chwilę klucz CLEAR w górę
3. Umieścić w rejestrze IC wartość zero: wybrać IC przełącznikiem, ustawić wszystkie klucze informacyjne na zero (w dolnym położeniu) i podnieść na chwilę do góry klucz LOAD
4. Uruchomić maszynę - ustawić klucz START w górę
5. Wybrać na kluczach informacyjnych numer systemowy końcówki (przy jednej końcówce - $\emptyset$, przy dwóch $\emptyset$ lub 1, przy czterech $\emptyset$, 1, 2 lub 3) i nacisnąć w górę klucz OPRQ na płycie czołowej maszyny. Końcówka, oczywiście, musi być załączona i sprawna. Po wykonaniu tej procedury system wyprowadzi na wybraną końcówkę komunikat

SET TIME

z którym należy postąpić w sposób wyżej opisany.

Przy prawidłowym zasilaniu maszyny i sprawnym procesorze uszkodzenia systemu operacyjnego zdarzają się niezwykle rzadko - niesprawność należy zazwyczaj szukać w urządzeniach peryferyjnych.

Jeśli jednak wyżej opisane czynności nie dają pozytywnego wyniku należy ściągnąć kopię systemu z dysku do pamięci maszyny, wykonując niżej podane czynności:

1. Zatrzymać maszynę podobnie jak to podano poprzednio
2. Załączyć czytnik i umieścić w nim taśmę z programem ładowania systemu (LOADER)
3. Wyzerować maszynę kluczem CLEAR
4. Umieścić w rejestrze IC wartości 0120276 (jedynki na bitach 0, 2, 8, 10, 11, 12, 13, 14), przy standardowej konfiguracji maszyny
5. Umieścić w rejestrze AR wartość $\emptyset$
6. Podnieść na chwilę klucz BIN, co spowoduje wczytanie programu ładującego. Po wczytaniu rejestr AR powinien zawierać wartość 0143. Jeśli nie, należy czynności powyżej opisane powtórzyć od punktu 2
7. Wyzerować maszynę kluczem CLEAR
8. Umieścić w rejestrze IC wartość $\emptyset$
9. Uruchomić maszynę ustawiając klucz START w górnym położeniu.

Wpisywanie się do systemu

W celu wpisania się do systemu należy w odpowiedzi na komunikat

WHO ARE YOU ?

wprowadzi z końcówki informację o postaci

<nazwa użytkownika> , <kod> .

gdzie <nazwa użytkownika> jest identyfikatorem (zwykle nazwiskiem) z którego system rozpoznaje 6 pierwszych znaków, zaś kod - trzema znakami sterującymi tj. znakami wprowadzanymi przy naciśniętym klawiszu CNTRL, niewidocznymi na wydruku czy na ekranie. Po rozpoznaniu, że nazwa i kod są prawidłowe, system zgłasza się podając swoją nazwę i oczekuje w nasłuchu na wprowadzenie zleceń z końcówki.

Niektóre zlecenia systemu operacyjnego

Zlecenia systemu CROOK mieszczą się w pojedynczych wierszach. Znakiem kończącym zlecenie i powodującym jego realizację jest kropka. Zlecenia dotyczące procesów (programów) są poprzedzane nazwą procesu (np. BASIC) i oddzielone od dwuliterowego kodu zlecenia dwukropkiem. Nazwę tę można pominąć, gdyż system pamięta nazwę procesu, do którego odnosiło się ostatnie takie zlecenie bądź też komunikat alarmowy dotyczący procesu. Jeżeli pominiemy nazwę procesu, a system nie zna jej z kontekstu - wtedy zlecenie będzie się odnosiło do procesu systemowego, obsługującego końcówkę użytkownika, który ma nazwę OSL.

Zlecenia nie dotyczące procesów rozpoczynają się zawsze dwuliterowym kodem zlecenia.

Błędy przy wprowadzaniu zleceń można korygować znakami sterującymi: n znaków BS kasuje n poprzedzających znaków, zaś znak CAN kasuje wszystko, co dotychczas wprowadzono w bieżącym wierszu.

Parametry, które są liczbami mogą być wprowadzane w postaci sum i/lub różnic dowolnej kombinacji liczb dziesiętnych i ósemkowych (oktalnych). Należy tylko pamiętać, że zarówno poszczególne składniki jak i wynikowy parametr muszą być liczbami całkowitymi 16-bitowymi, tzn. zawierać się w przedziale $-32768 \dots +32767$. Jako oktalną system traktuje liczbę rozpoczynającą się cyfrą zero.

Jedynym zleceniem, które nie ma dwuliterowego kodu jest zlecenie uruchomienia procesu od adresu początkowego (bądź utworzenia procesu i uruchomienia od adresu początkowego). Proces BASIC uruchamia się pisząc - jak to już uprzednio podawano:

BASIC.

lub też samą tylko kropkę, jeżeli system pamięta nazwę procesu z kontekstu - np. po alarmie.

Poniżej omówiono składnię niektórych innych zleceń systemu CROOK oraz wykonywane przez nie czynności.

CG[, [$\langle$ nazwa zbioru $\rangle$]].

Zlecenie bez parametru powoduje wydrukowanie informacji o zbiorach danego użytkownika. Drukowane są następujące informacje:

- znak oznaczający ochronę zbioru:
 - * zbiór trwały z ochroną przed zapisem i usunięciem
 - # zbiór trwały bez ochrony przed zapisem i usunięciem
 - bez znaku: zbiór nietrwały, usuwany po zakończeniu sesji.

- nazwa zbioru

- typ zbioru

CHR zbiór znakowy

ABS zbiór binarny (o dostępie swobodnym)

BAB zbiór binarny utworzony zleceniami DMP lub VRU

PRG zbiór binarny zawierający obraz pamięci programu
(niedostępny dla programów w języku BASIC)

- dwie liczby

Pierwsza z tych liczb oznacza zarezerwowaną przez system na zbiór liczbę sektorów (każdy sektor ma pojemność 256 słów 16-bitowych czyli 512 bajtów).

Druga liczba oznacza dla zbiorów znakowych liczbę sektorów zajmowanych przez informację zawartą w zbiorze, zaś dla zbiorów binarnych liczbę słów 16-bitowych zajmowanych przez informację zawartą w zbiorze. Druga liczba równa zero oznacza, że zbiór jest pusty.

Podanie zlecenia z parametrem powoduje wyprowadzenie informacji dotyczących tylko tego zbioru, którego nazwa jest parametrem zlecenia.

Oprócz informacji o zbiorach podawana jest reszta limitu sektorów.

Jeżeli wykonuje się po sobie kilka zleceń, dotyczących tego samego zbioru, można pominąć nazwę zbioru (ale nie przecinek !) w drugim i następnych zleceniach, gdyż system pamięta nazwę zbioru, którego dotyczyło ostatnie zlecenie.

$$CF, [\langle \text{nazwa zbioru} \rangle [, \left\{ \begin{array}{l} \text{CHR}, \langle \text{liczba sektorów} \rangle \\ \{ \text{ABS} \} [, \langle \text{liczba słów} \rangle] \\ \text{BAB} \end{array} \right\}]]] .$$

Zlecenie to służy do utworzenia nowego zbioru. Przy pracy w systemie BASIC jest ono rzadko potrzebne, gdyż system BASIC sam tworzy sobie zbiory - zleceniami SAVE, DMP, VRU, COM oraz instrukcją FILE.

Zbiory w systemie CROOK znajdują się zawsze na ciągłym, jednorodnym obszarze dysku i system musi przy tworzeniu zbioru znaleźć na dysku i zająć obszar zapewniający pomieszczenie całości informacji przeznaczonej do umieszczenia w zbiorze. Zbiory tworzone przez system BASIC rezerwują obszary o następujących wielkościach (z wyjątkiem zlecenia COM)

CHR 32 sektory
ABS 16 sektorów (4096 słów)

Jeżeli tak utworzony przez BASIC zbiór okaże się za mały, co objawia się przy wykonywaniu programu lub zlecenia wyprowadzeniem przez system komunikatu

BASIC END OF FILE <adres>

lub jeśli z góry wiemy, że jest potrzebny zbiór większy, należy taki zbiór utworzyć za pomocą opisywanego właśnie zlecenia. Przy pominięciu poszczególnych parametrów system przyjmuje:

- przy braku całego drugiego członu: zbiór CHR o wielkości 16 sektorów
- przy braku wielkości zbioru CHR: 16 sektorów
- przy braku wielkości zbioru ABS: 4096 (010000) słów
czyli 16 sektorów.

Próba utworzenia zbioru o takiej samej nazwie jak istniejący powoduje wydrukowanie tekstu OUR NAME.

TF, [<nazwa zbioru>] .

Zlecenie to powoduje obcięcie zbioru, tj. zwolnienie sektorów na końcu zbioru nie zajętych przez informację umieszczoną w zbiorze.

PF, [<nazwa zbioru>] [, [<nowa nazwa>] [, FUL]] .

Zlecenie służy do zmiany zbioru z nietrwałego na trwały, do zmiany nazwy zbioru i/lub do nałożenia ochrony przed zapisem i usunięciem. Zbiór wynikowy jest zawsze zbiorem trwałym, z ochroną przed zapisem i usunięciem. Zbiór będący przedmiotem operacji może być zbiorem nietrwałym albo zbiorem trwałym bez ochrony. Zlecenie wykonuje następujące czynności:

- jeśli zbiór jest nietrwały - zostaje przepisany (w systemie dwutalerzowym) na talerz użytkowy (wymienny) pamięci dyskowej i zamieniony na trwały;
- jeśli podano nową nazwę - ulega zmianie nazwa zbioru;
- nakłada się na zbiór ochronę przed zapisem i usunięciem;
- jeśli nie podano członu FUL, zbiór zostaje obcięty - tj. zostają zwolnione sektory nie zajęte przez informację w zbiorze.

Próba utrwalenia zbioru pustego powoduje wydrukowanie komunikatu EMPTY FILE.

Po pomyślnym wykonaniu zlecenia system podaje wartość reszty limitu sektorów. Jest to liczba sektorów, dostępnych jeszcze użytkownikowi na umieszczanie zbiorów trwałych. Wielkość zbioru, który chcemy utrwalić musi być zawsze mniejsza od tej liczby, w przeciwnym wypadku system wydrukuje przy próbie wykonania zlecenia PF komunikat LIMIT OF.

Należy w takim przypadku usunąć część niepotrzebnych zbiorów. Reszta limitu sektorów podawana jest również przy zleceniu CG.

NP, [<nazwa zbioru>].

Zlecenie powoduje zdjęcie ochrony przed zapisem i usunięciem dla wskazanego zbioru.

EF, [$\langle$ nazwa zbioru $\rangle$].

Zlecenie powoduje usunięcie wskazanego zbioru ze skorowidzów oraz zwolnienie zajmowanego przezeń obszaru dysku. Jeśli zbiór był zbiorem trwałym, to po wykonaniu zlecenia drukowana jest wartość reszty limitu sektorów, już powiększona w wyniku dodania liczby sektorów zajmowanych uprzednio przez usunięty zbiór.

Próba usunięcia zbioru z ochroną przed zapisem i usunięciem powoduje wyprowadzenie komunikatu FILE PROTECTED. Tak więc usunięcie zbioru z ochroną wymaga wydania dwóch zleceń, zlecenia NP i zlecenia EF, co daje nieco czasu na zastanowienie się i uniknięcie pomyłkowego usunięcia zbioru.

GF, [$\langle$ nazwa zbioru $\rangle$] , $\langle$ nazwa użytkownika $\rangle$.

Zlecenie umożliwia przekazanie zbioru innemu użytkownikowi. Żadne inne parametry zbioru nie zostają w wyniku wykonania tego zlecenia zmienione. Przekazanie zbioru nietrwałego jest możliwe tylko wtedy, gdy użytkownik któremu zbiór przekazujemy jest w chwili przekazywania wpisany do systemu (pracuje pod systemem).

AI , $\langle$ nr strumienia we $\rangle$ [, [$\langle$ nazwa zbioru $\rangle$]].

Zlecenie służy do utworzenia strumienia wejściowego, poprzez który będzie podawana informacja z danego zbioru znakowego. Jeżeli strumień o podanym numerze jest w chwili wykonywania zlecenia połączony z innym zbiorem, połączenie takie zostaje w wyniku zlecenia usunięte.

Wydanie zlecenia w postaci

AI , $\langle$ nr strumienia wejściowego $\rangle$.

powoduje ustawienie wskaźnika pobierania informacji na początek zbioru oraz wydrukowanie nazwy i wielkości zbioru związanego z danym strumieniem.

Zlecenia tego jak również zlecenia A0 należy używać przy pracy w systemie BASIC z rozważą, gdyż sam BASIC używa również strumieni do przesyłu informacji, definiując je ekstrakodami.

A0, <nr str.wy> [, [<nazwa zbioru>] [, <liczba sektorów>]] .

Zlecenie ma działanie podobne do opisanego wyżej zlecenia AI i służy do powiązania strumienia wyjściowego ze zbiorem znakowym, do którego ma być wyprowadzana informacja.

Podanie trzeciego parametru - liczby sektorów - powoduje utworzenie w wyniku wykonania zlecenia nowego zbioru znakowego o podanej liczbie sektorów.

Do zbiorów można poprzez tak utworzone strumienie wyprowadzać informację zleceniami LIST i NLI języka BASIC, poprzez podany w zleceniu za znakiem numeru # numer strumienia wyjściowego, jak również wyprowadzać i wprowadzać informację za pomocą innych programów systemu CROOK.

TM.

Zlecenie to powoduje wydrukowanie przez system czasu bieżącego w godzinach, minutach i sekundach.

DT.

Zlecenie powoduje wydrukowanie na końcówce daty.

ES.

Zlecenie to powoduje zakończenie sesji.

Zbiory nietrwałe oraz procesy są usuwane, poczem system drukuje komunikat

WHO ARE YOU ?

Oczekując na wpisanie się następnego użytkownika.

FS.

Działa tak samo, jak zlecenie ES., lecz nie usuwa zbiorów nietrwałych.

BASIC : RS.

Wznowienie działania procesu po zatrzymaniu, np. po przerwaniu pracy, po alarmie dotyczącym urządzenia i usunięciu niesprawności urządzenia (patrz niżej).

BASIC : DE.

Usunięcie procesu z systemu.

Zatrzymywanie i wznowianie pracy systemu BASIC

Zdarza się, że zachodzi potrzeba zatrzymania systemu BASIC, np. jeśli w czasie wykonywania programu i wyprowadzania wyników widać, że wyniki są złe, gdy program działa za długo w pętli i są podejrzenia, że jest w nim błąd gdy zapłaczę się taśma w drukarce itp. albo gdy z jakichkolwiek innych powodów chcemy przejść na nasłuch systemu operacyjnego. Dokonujemy tego generując przerwanie - naciskając klawisz BREAK na klawiaturze (albo, jeśli końcówka nie jest wyposażona w ten mechanizm - klucz OPRQ na płycie czołowej maszyny przy ustawionym jednocześnie na kluczach informacyjnych systemowym numerze końcówki).

Przyjęcie przez maszynę przerwania powoduje zawsze przejście na nasłuch systemu operacyjnego, co jest sygnalizowane wyprowadzeniem na końcówkę nowego wiersza. Wykonujący się w czasie generacji przerwania proces (w naszym przypadku zazwyczaj BASIC) zostanie zatrzymany tylko wtedy, jeśli zajmował strumień wejściowy lub wyjściowy zero (tj. końcówkę). Zostanie to zasygnalizowane tekstem

BASIC STOPPED

Jeżeli system BASIC pracuje nadal po przerwaniu, tj. jeżeli nie został wyprowadzony powyższy komunikat, można go zatrzymać zleceniem:

BASIC : SP.

Po zatrzymaniu BASIC-u system operacyjny jest na nasłuchu i można wykonywać zlecenia systemu operacyjnego.

Wznowienie pracy BASIC-u od stanu, w którym została przerwana można uzyskać wprowadzając zlecenie

BASIC : RS.

Jak już mówiliśmy, system pamięta nazwę programu, którego dotyczyło ostatnie zlecenie lub komunikat o błędzie. Można sobie zatem oszczędzić czasu pomijając tę nazwę. I tak, na przykład, w wyżej opisanym przypadku zamiast pisać

BASIC : RS.

można zlecenie skrócić do postaci:

RS.

Niektóre komunikaty systemu operacyjnego

Poniżej podano znaczenia tekstów niektórych komunikatów drukowanych przez system operacyjny CROOK przy wystąpieniu błędów.

BASIC WRONG ADDRESS

BASIC WRONG INSTR.

Wystąpienie tych błędów oznacza zazwyczaj zniszczenie części systemu BASIC, które może wystąpić z różnych powodów. Jeżeli błąd wystąpił w sytuacji, gdy mamy w pamięci w zbiorze roboczym program, który nie jest zapamiętany na dysku, należy spróbować umieścić go na dysku (co jest bardzo często możliwe mimo uszkodzenia BASIC-u), wykonując następujące czynności:

1. Wywołać system BASIC, wprowadzając zlecenie

BASIC.

lub samą tylko kropkę. System BASIC powinien zgłosić się

znakiem nasłuchu * lub # . Jeżeli wykonanie tej czynności powoduje wydrukowanie przez system operacyjny jednego z opisanych komunikatów alarmowych - nic z treścią zbioru roboczego nie da się zrobić i należy przejść do punktu 3.

2. Zapamiętać program na dysku zleceniem SAVE (zmieniwszy wpierw nazwę zleceniem PROG). Przy wyniku negatywnym przejść do punktu 3 - program jest niestety nie do odzyskania. Na szczęście błędy takie występują niezwykle rzadko.
3. Po zapamiętaniu na dysku treści zbioru roboczego albo przy negatywnym wyniku prób, należy usunąć starą, a ściągnąć z dysku nową kopię systemu BASIC, wykonując kolejno zlecenia:

BASIC : DE.

BASIC.

po wykonaniu których nowa kopia systemu BASIC powinna się zgłosić na końcówce znakiem nasłuchu.

BASIC DEVICE <nr fizyczny urządzenia>

Niesprawne urządzenie wejścia/wyjścia - np. brak taśmy w czytniku, drukarka w stanie off-line itp. Należy usunąć niesprawność i wznowić działanie BASIC-u zleceniem RS.

Poniżej podano dla orientacji przyporządkowanie numerów i nazw do odpowiednich urządzeń w standardowym zestawie maszyny MERA 400.

Drukarka mozaikowa DZM180:

nr fizyczny : 5

nr strumienia wyjściowego CROOK : 2

nazwa zbioru publ.BASIC : "LP2***"

Dziurkarka taśmy papierowej:

nr fizyczny : 6

nr strumienia wyjściowego CROOK : 1

nazwa zbioru publ.BASIC : "PUN***"

Czytnik taśmy papierowej:

nr fizyczny : 5

nr str.wejściowego CROOK : 1

nazwa zbioru publ. BASIC : "TR1***"

Końcówki użytkowników (np. DZM-180 KSR , MERA 9452) mają numery fizyczne 4, 3, 2, 1. Numerem strumienia dla końcówki jest zawsze 0 (tak dla wejścia, jak i dla wyjścia) , zaś nazwami w BASIC-u: "MOI***" dla wejścia i "MOO***" dla wyjścia.

BASIC PARITY <nr fizyczny urządzenia>

Błąd parzystości znaku wprowadzanego z urządzenia wejściowego. W zależności od sytuacji można próbować wznowić działanie BASIC-u zleceniem RS bądź od nowa wywołać BASIC i uruchomić program (starając się najpierw ustalić przyczynę powstania błędu).

BASIC DISC ERROR <kod błędu>

Niesprawność dysku. Kod błędu 0140 oznacza, że dysk nie jest rozpędzony i nie osiągnął stanu roboczego. Należy wtedy odczekać pewien czas i ponownie wykonać operację, która spowodowała wydruk alarmowy. Inne błędy dysku należy zgłosić kompetentnej osobie, a nie starać się ich poprawiać - gdyż można w ten sposób wprowadzić trwałe uszkodzenia do systemu zbiorów.

BASIC END OF FILE

Próba zapisu poza obszar zarezerwowany na zbiór. Należy zbiór usunąć, utworzyć nowy, większy zleceniem CF i wykonać program jeszcze raz.

BASIC FINISH <adres>

Komunikat ten oznacza prawidłowe przejście z BASIC-u do systemu operacyjnego, dokonane zleceniem BYE.

Poniższe komunikaty mogą wystąpić w czasie wykonywania zleceń systemu operacyjnego

TABLE OF

Przepełnienie tablic systemu operacyjnego (przy próbie utworzenia procesu). Należy zleceniem DE usunąć któryś z niepotrzebnych procesów, by zwolnić zajmowane przez nie miejsce w tablicach systemu.

WRONG DIGIT

Cyfra 8 lub 9 w liczbie oktalnej. Należy zlecenie wprowadzić powtórnie w poprawnej postaci.

OVERFLOW

Nadmiar liczby w parametrze. Liczba taka nie może być większa niż 32767 ani mniejsza niż -32768. Maksymalnym adresem jest 0177777, co odpowiada liczbie dziesiętnej - 1.

ER COM

Błąd w dwuliterowym kodzie zlecenia (kod zlecenia nie został rozpoznany).

COMMAND

Próba zatrzymania procesu (programu) zatrzymanego lub wznowienia działania procesu działającego.

UNKNOWN FILE

System nie znalazł w skorowidzach zbioru o podanej nazwie. Na ogół skutek popełnienia błędu w nazwie zbioru.

LIMIT OF

Próba utrwalenia zbioru (zamiany zbioru z nietrwałego w trwały zleceniem PF), która powodowałaby przekroczenie limitu - patrz opis zlecenia PF.

UNKNOWN USER

Próba przekazania zleceniem GF zbioru użytkownikowi nie istniejącemu w systemie, lub nie wpisanemu w danej chwili - przy przekazywaniu zbioru nietrwałego. Na ogół skutek popełnienia błędu w nazwie użytkownika.

OUR NAME

Próba utworzenia zleceniem CF zbioru o takiej samej nazwie jak już istniejący.

ENGAGED

Występuje przy wpisywaniu się do systemu w odpowiedzi na komunikat WHO ARE YOU ? i oznacza próbę wpisania się z innej końcówki użytkownika, który już pracuje na innej końcówce.

W powyższym zestawieniu podano jedynie te zlecenia i komunikaty systemu, które są potrzebne i występują najczęściej przy pracy w BASIC-u. Szersze informacje są podane w podręczniku systemu operacyjnego CROOK.

Niektóre programy codziennego użytku

Biblioteka programów systemu CROOK zawiera kilka programów, które mogą być bardzo przydatne użytkownikowi BASIC-u. Poniżej podano sposób wywołania tych programów oraz krótki opis ich przeznaczenia.

CAT, <nr strumienia wyjściowego> .

Program drukuje na urządzeniu powiązanym z podanym strumieniem wyjściowym katalog zbiorów użytkownika - podobnie jak zlecenie CG , lecz bez zbiorów typu PRG zawierających obrazy pamięci procesów.

Wywołanie tego programu w postaci

CAT,Ø,3.

powoduje wydrukowanie na końcówce nazwy aktualnie wpisanego użytkownika.

STAPE, <nr strum.we1> , <nr strum.we2> .

Program porównuje wierszami informację z obu strumieni i wyprowadza na końcówkę treść wierszy różniących się między sobą. Może służyć do sprawdzania zgodności taśmy papierowej z zawartością zbioru itp.

COPY, <nr str.we> , <nr str.wy> [, 1].

Program kopiuje znak po znaku informację ze strumienia wejściowego na strumień wyjściowy, aż do napotkania znacznika końca zbioru (znaku DC4 w nowym wierszu). Znacznik końca zbioru jest wyprowadzany do strumienia wyjściowego, chyba że podano trzeci parametr - wtedy nie jest wyprowadzany.

LIST [, [<nr str.wy>] [, <nr str. we>]].

Najczęściej używaną postacią uruchamiania tego programu jest postać bez parametrów:

LIST.

po której program wyprowadza znak zapytania. Należy wtedy wprowadzić nazwę zbioru i zakończyć ją znakiem LF. Program wyprowadzi wówczas zawartość tego zbioru wierszami na końcówkę, a po napotkaniu wskaźnika końca zbioru w strumieniu wejściowym wyprowadza na końcu znak nowej strony FF.

Podanie pierwszego parametru - nr strumienia wyjściowego, różnego od zera, powoduje wyprowadzanie na wskazany strumień, z podziałem na strony. Podanie drugiego parametru powoduje pobieranie informacji nie ze zbioru, a ze wskazanego strumienia. Program również wtedy wyprowadza po uruchomieniu znak zapytania i podaną informację zakończoną znakiem LF traktuje jako nagłówek.

LISM.

Wersja programu LIST przeznaczona głównie do drukowania treści zbiorów na monitorach ekranowych MERA-9452 używanych jako końcówki. Po każdorazowym zapełnieniu ekranu program zatrzymuje się. Chcąc kontynuować, należy wprowadzić znak CAN (naciśnięty klawisz CNTRL i jednocześnie klawisz X) lub LF, natomiast chcąc zakończyć wykonywanie programu należy wprowadzić znak DC4 (naciśnięty klawisz CNTRL i jednocześnie klawisz T). Stronicowanie i wyprowadzanie znaku FF na końcu nie jest wykonywane.

DLIST , <nr str.wy> [, <nr str.we>] .

Program jest przeznaczony do wyprowadzania informacji na drukarkę o szerokim papierze. Wydruk podzielony jest na strony opatrzone numerem strony, oraz datą i nazwą drukowanego zbioru w taki sposób, że nieparzyste strony są na lewych połówkach papieru, a parzyste na prawych. Wydruk taki po rozdzieleniu arkuszy i złożeniu ich na pół można wpiąć do skoroszytu czy też oprawić. Znaczenie parametrów jest takie same jak w programie LIST.

Do podobnych celów służy napisany w BASIC-u program DLISTB, przeznaczony do drukowania programów w języku BASIC, umieszczonych w zbiorach zleceniem SAVE. Wykonuje on dodatkowo: rozpoczynanie od nowej strony subprogramów (przy napotkaniu napisu NAME w nowym wierszu); wcinanie wewnątrz bloków pętli FOR-NEXT i wcinanie wewnątrz bloków definicji.

Wyżej opisane programy codziennego użytku systemu CROOK są programami pomocniczymi. Po zakończeniu wykonywania takiego programu odpowiadający mu proces jest automatycznie usuwany z tablic systemowych, co system operacyjny oznajmia komunikatem informacyjnym:

<nazwa procesu> DELETED

WERSJA BEZDYSKOWA SYSTEMU BASIC

Aby umożliwić pracę w języku BASIC również tym użytkownikom, którzy mają maszynę MERA-400 bez pamięci dyskowej MERA 9425, opracowano uproszczoną wersję opisanego tu języka, nie realizującą pewnych mechanizmów. Konieczne było wprowadzenie ograniczeń ze względu na niedużą objętość pamięci - po to, by dało się w systemie wykonywać programy o rozsądnej wielkości. Obszar pamięci na program można powiększyć rezygnując z pewnych mechanizmów języka przy wywołaniu systemu - co będzie opisane dalej.

Najważniejsze ograniczenia wersji bezdyskowej języka BASIC są następujące:

1. System nie realizuje odwracania macierzy, transpozycji macierzy oraz mnożenia zmiennych indeksowanych.
2. Nie są rozpoznawane niektóre funkcje: JOB\$, CUF\$, USR\$, SYS\$; funkcja DAT\$ ma wartość łańcucha pustego.
3. Nie można, z powodu braku dysku, wykonywać żadnych operacji na zbiorach o dostępie swobodnym.
4. Nie są rozpoznawane zlecenia związane z dyskiem, takie jak: SAVE, OLD, DMP, VRU, LGO, COM, NSAVE oraz także instrukcje, np. instrukcja LIBRARY.
5. Możliwe jest używanie instrukcji CHAIN, ale nowy program może być, oczywiście, ściągany tylko z urządzenia zewnętrznego, np. czytnika taśmy papierowej:

```
1500 CHAIN "TR1***";
```

Zlecenie o postaci

MCA

powoduje wyprowadzenie przez BASIC komunikatu

CORE SIZE ?

po którym należy podać największy adres, który będzie dostępny systemowi BASIC. Pozwala to na umieszczenie w pamięci innych jeszcze oprócz BASIC-u programów, w obszarze pamięci o adresach

wyższych od podanego w zleceniu MCA.

Jeżeli systemowi BASIC zabraknie pamięci, wydrukuje komunikat

CORE SIZE ?

po którym należy mu w celu kontynuacji pracy podać adres większy od ustalonego poprzednio.

Dla ułatwienia komunikacji z innymi programami poprzez strumienie we/wy, wprowadzono w tej wersji dodatkowe nazwy zbiorów publicznych: "IN1***", "IN2***", ..., "IN7***" do komunikacji poprzez strumienie wejściowe odpowiednio 1,2,...,7 oraz "OU1***", "OU2***", ..., "OU7***" do komunikacji poprzez strumienie wyjściowe odpowiednio 1, 2, ..., 7.

Zwiększenie pamięci na program kosztem rezygnacji z niektórych możliwości można uzyskać uruchamiając BASIC z podaniem ujemnego parametru. Poniżej podano zlecenia wywołania i niedostępne przy nich operacje (każda dalsza pozycja oznacza, że nie ma wtedy w BASIC-u wszystkich możliwości wymienionych powyżej oraz w danej pozycji).

BASIC, -1.

operacje na zmiennych indeksowanych MAT

BASIC, -2.

operacje na wartościach łańcuchowych. Można używać jedynie stałych łańcuchowych w instrukcjach drukowania

BASIC, -3.

drukowanie według wzorca

BASIC, -4.

instrukcja zmiany programu (nakładkowania) CHAIN.

2PPG/426, /81/200

E R R A T A
 =====

str./wiersz	jest	ma być
15 ²	$1 - \sin^2 x$	$\sqrt{1 - \sin^2 x}$
25 ¹³	$\left\{ \begin{array}{l} \langle \text{instr. zatrzymania} \rangle \\ \langle \text{instrukcja skoku} \rangle \\ \langle \text{instr. sk. ze śladem} \rangle \\ \langle \text{instr. powrotu wg śladu} \rangle \end{array} \right\}$	$\left\{ \begin{array}{l} \langle \text{instr. zatrzymania} \rangle \\ \langle \text{instrukcja skoku} \rangle \\ \langle \text{instr. sk. ze śladem} \rangle \\ \langle \text{instr. powr. wg śladu} \rangle \end{array} \right\}$
27 ²	Obliczanie	Obliczenia
27 ³	zapamiętania	zapamiętanie
85 ⁵	indekсы	(indexy)
93 ₁₄	wynosi "ABCABCABC"	ma wartość "ABCABCABC"
96 ₁₆	znak numeru	# znak numeru
98 ¹⁵	program BASIC	system BASIC
102 ₁₃	do wiersza o największym numerze	(do wiersza o największym numerze)
104 ₁₆	jak zlecenie	jak w zleceniu
108 ¹¹	assemblerze	(assemblerze)
109 ₁₆	CLK	CLOCK
111 ₁₄	po pktcie 9. dodać: 10. Wykonać czynności pktu 5 z poprzedniej strony.	
111 ₉	wprowadzi	wprowadzić
113 ¹³	pamięci programu	pamięci procesu
114 ⁶	Zbiory w systemie CROOK znajdują się	Zbiór w systemie CROOK znajduje się
116 ₆	wyniku zlecenia	wyniku wykonania zlecenia
118 ¹¹	w nim błąd	w nim błąd,
119 _{8,7}	w zbiorze roboczym	(w zbiorze roboczym)
127 ¹²	strumienie wejściowe	strumienie wyjściowe